AZƏRBAYCAN RESPUBLİKASI
ELM VƏ TƏHSİL NAZİRLİYİ

Azərbaycan Kibertəhlükəsizlik
Təşkilatları Assosiasiyası

Veb təhlükəsizlik

fənni üzrə mühazirələr toplusu

Adil Əliyev, Nurtəkin Quliyeva, Rəşad Vəliyev, Zəkiyə Əlizadə

Bu vəsait Azərbaycan Respublikasının Elm və Təhsil Nazirliyinin maliyyə dəstəyi ilə həyata keçirilən "İnformasiya təhlükəsizliyi ixtisası üzrə elmi-metodiki tədris (çap və onlayn) vəsaitlərinin hazırlanması" layihəsi çərçivəsində hazırlanmışdır.

Nəşrin məzmununa görə donor məsuliyyət daşımır.

Bakı – 2024.

Hazırladılar: Adil Əliyev
Nurtəkin Quliyeva
Rəşad Vəliyev
Zəkiyə Əlizadə

İnformasiya Təhlükəsizliyi sahəsində kadr potensialının gücləndirilməsi məqsədilə 2022-ci ildə Elm və Təhsil Nazirliyi və Dövlət Təhlükəsizliyi Xidmətinin birgə təşəbbüsü ilə ölkənin aparıcı ali təhsil müəssisələrinin və müxtəlif dövlət qurumlarının nümayəndələri ilə yanaşı, dünyanın nüfuzlu universitet və şirkətlərində çalışan azərbaycanlı mütəxəssislər, o cümlədən AKTA sədri Elvin Balacanov və hazırda Samsung Electronics şirkətində "Knox" mobil təhlükəsizlik həlləri üzərində çalışan və bu mühazirələr toplusunun həmmüəllifi Adil Əliyevin iştirakı ilə hazırlanmış təhsil proqramı elm və təhsil nazirinin 2022-ci il 28 iyul əmri ilə təsdiqlənmişdir.

Proqramda yer almış bir sıra fənnlərin tədrisinə dəstək məqsədi ilə nümunəvi sillabuslar hazırlanıb ali məktəblərə təqdim edilmişdir. Həmin təşəbbüsün davamı olaraq Təhsil Nazirliyinin maliyyə dəstəyi ilə "İnformasiya təhlükəsizliyi ixtisası üzrə elmi-metodiki tədris (çap və onlayn) vəsaitlərinin hazırlanması" layihəsi çərçivəsində "Veb təhlükəsizlik" fənni üzrə mühazirələr toplusunu hazırlayıb sizlərə təqdim edirik.

Bu mühazirələr müəlliflər tərəfindən ilk dəfə olaraq 2022-ci ilin payız semestrində Azərbaycan Texniki Universitetində tədris edilmişdir. Dərs prosesindəki toplanmış təcrübə və tələbələrin rəyləri əsasında sillabusda redaktələr edilmiş və yekun versiya hazırlanmışdır. Həmçinin, semestr boyu dərslərdə fəallıq göstərmiş tələbələr Nurtəkin Quliyeva və Rəşad Vəliyev bu mühazirələrin yenidən işlənməsində yaxından iştirak edərək həmmüəllif olmuşlar. AzTU-nun bu proseslərdə mühüm rol oynadığına görə universitet rəhbərliyinə, "Kibertəhlükəsizlik" kafedrasının əməkdaşlarına və xüsusilə kafedranın rəhbəri t.e.d, dosent Yadigar İmamverdiyevə minnətdarlığımızı bildiririk.

Materiallar hazır olduqca ilkin versiyaları gözdən keçirib rəylər verməklə Tofiq Hüseynov və Rəşad Əliyev bu mühazirələrin praktiki əhəmiyyətinin artırılmasında dəstək olduqları üçün təşəkkür edirik.

Materiallarla bağlı hər-hansı bir sual və ya qeyd olarsa adil@ozunoyren.com ünvanına yaza bilərsiniz.

İçindəkilər

1 | Mühazirə 1

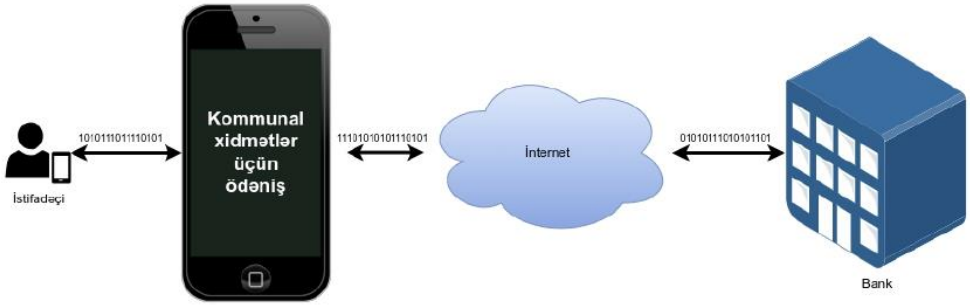
Giriş

Veb təhlükəsizlik nədir?

Veb təhlükəsizlik veb proqram təminatlarını kiber-təhdidlərdən qorumağı öyrənən bir sahədir. Veb təhlükəsizliyin əsas məqsədi veb proqram təminatlarının fasiləsiz şəkildə işləməsini təmin etmək və həmin proqramlara qarşı yönəlmiş bir sıra hücumların qarşısını almaqdır.

Bu gün demək olar ki, hər kəs internetdən istifadə edir. Veb saytlar, istər mobil istərsə də smart televizorlardakı tətbiqlər, soyuducularda quraşdırılmış proqram təminatları, avtomobillər və bizi əhatə edən bir çox qurğular internet vasitəsilə məlumat mübadiləsi həyata keçirir.

Bir çoxumuz bank əməliyyatlarını mobil tətbiqlər vasitəsilə yerinə yetiririk. Məsələn, komunal xidmət üçün bankın mobil tətbiqi vasitəsilə ödəniş etdikdə, mobil tətbiqdən məlumat sorğusu banka göndərilir. Daha sonra bankın sistemləri göndərilən məlumatı emal edir və lazım olan əməliyyatları yerinə yetirir.



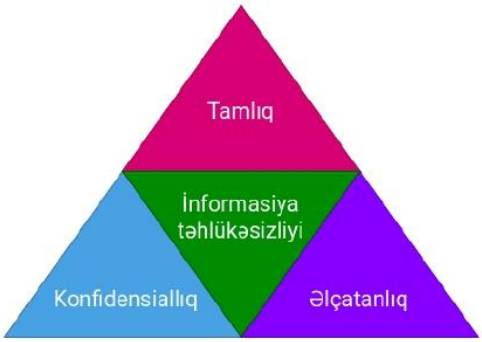
Şəkil 1.1: Mobil tətbiq və bank sistemi arasında tipik əlaqə

Bu zaman ötürülən məlumatlarların **konfidensial**, **tam** şəkildə və düzgün ünvana yəni banka **vaxtında çatdırılması** vacibdir. Diqqət etdinizsə əvvəlki cümlədə üç prinsipi sadalamış olduq. Bunlar informasiyanın **konfidensiallığı**, **tamlığı** və **əlçatanlığı**dır. Bu üç prinsip informasiya təhlükəsizliyinin özəyini təşkil edir.

Fərz edək ki, bir öncəki nümunədə dediyimiz bank tətbiqinin proqramçısısınız.Və yaxud elektron xidmət göstərən dövlət qurumunun veb saytını hazırlayırsınız. Sizcə bu sistemlərin təhlükəsizliyini təmin etmək üçün nə etmək lazımdır? Ümumiyyətlə hansı növ təhlükələr mövcuddur? Onlardan necə qorunmaq olar?

Bu kursun vəzifəsi yuxarıdakı sualların cavabını tapmaq, habelə veb təhlükəsizliyin nə olduğunu aydınlaşdırmaq, müxtəlif növ təhlükələr barədə məlumat vermək, bu təhlükə-

lərlə əlaqəli boşluqları aşkar etmək, öyrənmək məqsədilə sistemləri sındırmaq və onların müdafiə üsullarını öyrənməkdir. Təbii ki, bir sistemi qoruya bilmək üçün onun təhlükəsizliyinə həm müdafiə həm də hücum perspektivindən baxa bilmək olduqca vacibdir. Ona görə də kursda mövzularla bağlı anlayışlar verilir, daha sonra təhlükəsizlik boşluqları və onlardan istifadə edərək sistemlərin sındırılması və qorunması izah edilir.



Şəkil 1.2: İnformasiya təhlükəsizliyinin əsas üç prinsipi

Etika kodeksi

Əsas mövzulara başlamamışdan əvvəl oxuculara bu məlumatı vermək vacib hesab edirik. Əvvəl də qeyd etdiyimiz kimi, istənilən bir sistemi qoruya bilmək üçün onun təhlükəsizliyinə həm müdafiə həm də hücum perspektivindən baxa bilmək olduqca vacibdir. Yəni biz həm müdafiə ,həm də hücüm üsulları barədə məlumat verəcəyik.

Xüsusi olaraq qeyd etmək lazımdır ki, əldə edəcəyiniz bilik və bacarıqları qəti şəkildə qeyri qanuni və zərərli məqsədlər üçün istifadə etməməlisiniz. Öyrəndiklərinizi qanundan kənar məqsələr üçün istifadə edərsinizsə, müvafiq qanunvericilik üzrə məsuliyyət daşımış olacaqsınız.

Əminik ki, gələcəyin peşəkarları olaraq, bilik və bacarıqlarınızı yalnız rəqəmsal sistemləri qorumaq və yalnız cəmiyyətimizə tövhə vermək üçün istifadə edəcəksiniz.

Kursu uğurlu qavramaq üçün nə lazımdır?

Hər bir kursun özünəməxsus xüsusiyyətləri vardır. Kursun materiallarını yaxşı qavramaq üçün həm bu xüsusiyyətləri anlamaq həm də kursun məqsədini, tətbiq nöqtələrini öyrənmək vacibdir.

“Veb təhlükəsizlik” kursunda uğurlu olmaq üçün ilk olaraq kursun məqsədini başa düşmək lazımdır. Kursun məqsədi əvvəlki bölmədə verilmişdir. “Veb təhlükəsizlik” kursunun xüsusiyyətlərinə gəldikdə, burada əsas məsələ təhdidlərin növlərini başa düşmək və onlara qarşı preventiv (qabaqlayıcı) tədbirlər görməkdən ibarətdir.

Bəs bir sistemi qorumağı necə öyrənmək olar? Hər hansı bir sistemi qorumaq üçün əvvəl-

cə onu nədən qoruyacağımızı bilməliyik. Və onu qoruya bilmək üçün nə cür hücumların gələ biləcəyi barədə məlumatlı olmalıyıq. Bu o mənaya gəlir ki, müdafiəni uğurlu təşkil etmək üçün hücumu da bilmək lazımdır.

Şəkil 1.3: Haker kimi düşünmə “Veb təhlükəsizlik”də vacibdir.

Veb təhlükəsizliyi yaxşı qavramaq üçün siz hakerlər kimi düşünməyi bacarmalı və sistemlərdəki boşluqları istifadə etməklə onları sındırmağı bacarmalısınız. Daha sonra o boşluqların qarşısının necə alınmasını öyrənməklə müdafiəni təşkil etməyi öyrənməlisiniz.

Mövzunun aktuallığı

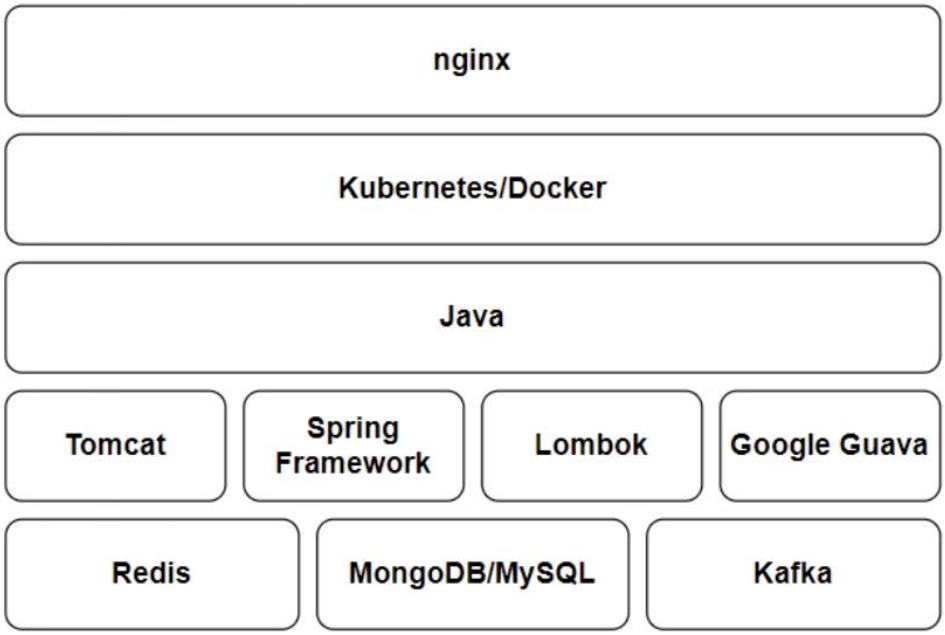
Kurs boyu bəzi hücum növlərinin (məsələn, XSS, SQL İnjection, MiTM, fişinq və s.) heç də yeni olmadığının şahidi olacaqsınız. Bir çox hücum növləri 90-cı illərdən, 2000-lərin əvvəllərindən mütəxəssislərə tanışdır. Bəs necə olur ki, indiyədək bu problemləri birdəfəlik aradan qaldırmaq mümkün olmayıb?

Məsələyə iki bucaqdan baxmaq olar. Bir tərəfdən sistemləri yaradanlar bucağından, digər tərəfdən isə sistemə hücum etmək istəyənlər bucağından. Heç də hər bir şirkətin güclü mütəxəssisləri işə götürmək imkanları yoxdur, amma ayaqda qalmaq üçün onlar layihələr yerinə yetirməlidirlər və pul qazanmalıdırlar. Belə olduqda şirkətlər sərištəsiz işçiləri qəbul etməyə məcbur olurlar. Tez-tez həmin işçilərə sistemlər yaratmaq həvalə olunur. Onlar isə təhlükəsizlik tədbirlərini ciddi şəkildə qurmağı bacarmır.

Amma təhlükəsizlik problemləri yalnız "yoxsul" şirkətlərin yaratdıqları sistemlərdə baş vermir. Bəzi hallarda isə güclü mühəndislər də təhlükəsizlik boşluqları buraxırlar. İnsanlar tez-tez səhv edə bilirlər. Bir çox hallarda onlar bilmədən və ya diqqətsizlik ucbatından xətalı kodlar yaza bilir, şəbəkəni qurarkən unutduqları məqamlar ola bilir. Elə hakerlərə də bu lazımdır.

Elə hallar da olur ki, təhlükəsizlik məsələlərini doğru şəkildə təşkil etməyə zaman az olur. Şirkət rəhbərliyi isə layihəni ən qısa zamanda bitirməyi tələb edir və bu halda təhlükəsizlik tədbirləri ikinci plana düşür. Bəzi hallarda isə işçilər bilərəkdən səhlənkarlıq edə bilirlər.

Bütün bu səbəblər təhlükəsizlik boşluqlarının peyda olmasına gətirib çıxarır. Amma iş burada bitmir. Təsəvvür edin ki, bir şirkətin sahibisiniz və sizin üçün ən prioritet məsələ kibertəhlükəsizlikdir. Ən yaxşı işçiləri götürmüsünüz, ən güclü tədbirləri görürsünüz. İstənilən halda sizin istifadə etdiyiniz texnologiyaların heç də hamısını özünüz yaratmırsınız. Məsələn, Java texnologiyası ilə bir sistem yaratmısınız. Müasir dövrdə bu arxitekturaya böyük ehtimal ki, aşağıdakılar mütləq daxildir.



Şəkil 1.4: Java texnologiyası ilə yaradılan bir sistem arxitekturasına nümunə

Yuxarıdakı diaqramda hər bir proqram təminatının özü də yüzlərlə fərqli kitabxanalardan ibarətdir. Sizin yazdığınız kod yalnız bu arxitekturanın kiçik bir hissəsidir. Siz öz kodlarınıza cavabdeh olsanız da nə bu diaqramdakı sistemlərə nə də onların tərkib hissələri olan kitabxanalara tam zəmanət verə bilməzsiniz. Bu o deməkdir ki, o sistemlərdə boşluq olarsa sizin yaratdığınız sistemdə də o boşluq olacaqdır. Məhz bu səbəbdən veb təhlükəsizliyin tərkib hissələrini bilmək mütləqdir. Digər tərəfdən isə hakerlər də daim sistemləri sındırmaqda maraqlıdırlar. Çünki, bəzi bədniyyətli insanlar bu yollarla pullar qazanırlar. Dünyadakı siyasi hadisələrdə kibermüharibələrin payı böyükdür. İndiki dövrdə məlumatlar kompüter şəbəkələri vasitəsilə əldə olunur və ötürülür. Ölkələr arasında kibermüharibələr mövcuddur, böyük rəqib şirkətlər arasında da kibermüharibələrin rolu böyükdür. Bir çox cinayətkarlar müasir dövrdə cinayət əməllərini internet üzərindən həyata keçirirlər. Kriptovalyutalar, blokçeyn texnologiyaları maraqlı olsalar da ,onlardan qeyri-qanuni işlərin ödənişləri üçün geniş istifadə edirlər.

Yuxarıda deyilənləri nəzərə alsaq kibertəhlükəsizlik mütəxəssisi kimi siz yalnız texnologiyanı deyil, dünya iqtisadiyyatı, dünya siyasəti, qanunvericilik barədə də məlumatlı olmalısınız.

Son bir neçə ildə baş vermiş bəzi kiberhücumlar

Rusiya-Ukrayna müharibəsi iki ölkə arasında həm də kibermüharibələrə səbəb oldu. 2022-ci ilin sentyabr ayında birdən-birə Rusiyanın paytaxtı Moskvanın mərkəzi küçələrinin biri taksilərlə dolmağa başladı və həmin bölgədə nəqliyyat hərəkəti iflic oldu. Səbəb Yandex Taksi xidmətinə olan kiberhücum idi. Ehtimala görə hücum Ukrayna hakerləri tərəfindən olmuşdur. Hakerlər Yandex Taksi sistemini sındıraraq bütün taksiləri eyni ünvana sifariş etməyi bacarmışdılar. Ünvanda isə heç kim gözləmirdi. Nəticədə taksilər heç bir müştəri əldə edə bilmədiklərindən bir qədər pul itirdilər. Şəhərin həmin bölgəsində hərəkət iflic vəziyyətə gəldiyindən bir çox fəsadlar yaşandı. Həmçinin Yandex Taksi şirkətinin nüfuzuna zərər dəydi [1].

Başqa bir nümunə kimi ölkəmizdə MİDA yaşayış kompleksində evlərin satış prosesində baş vermiş problemi göstərmək olar. "Güzəştli mənzil" sistemi vasitəsilə satışın təşkili agentliyin veb saytı vasitəsilə yerinə yetirilmişdi. Dələduzlar veb saytdan qeyri-qanuni şəkildə istifadə edilmək məqsədilə xüsusi zərərverici proqram hazırlamışlar. Nəticədə evlər bir neçə saniyə ərzində onlar tərəfindən "qazanılmışdı". Bu hücum sosial evlərdən yararlanmaq istəyən yüzlərlə vətəndaşın haqqını əlindən almışdı. Sonra Dövlət Təhlükəsizliyi Xidməti tərəfindən araşdırılma aparılaraq məsələnin üstü açılmış, günahkar şəxslər həbs olunmuş və nəticələr ləğv edilmişdi [2].

2017-2018-ci ildə Azərbaycanın bir sıra dövlət qurumlarına "MuddyWater" adlı haker qruplaşması tərəfindən hücumlar olmuşdur [3].

Mütəmadi şəkildə fərqli qurumların informasiya sistemlərində aşkar edilmiş boşluqlara dair hesabatların rəsmi qurumlar tərəfindən dərc edildiyinin də şahidi oluruq [4].

Bütün bunlar ondan xəbər verir ki, günümüzdə veb təhlükəsizlik boşluqları mövcud olmaqdadır və təhlükəsizlik alətləri nə qədər inkişaf etsə də boşluqların tam aradan qaldırılması mümkün deyildir. Belə olduqda veb təhlükəsizliyi yaxşı öyrənmək, həm hücum mexanizmlərini bilmək, həm də hücumlardan qorunmağı bacarmaq olduqca vacibdir.

---

[1] Moskova'yı karıştıran olay: Uygulama hacklenince bütün taksiler aynı noktaya hareket etti! (https://www.cnnturk.com/dunya/moskovayi-karistiran-olay-uygulama-hacklenince-butun-taksiler-ayni-noktaya-hareket-etti)

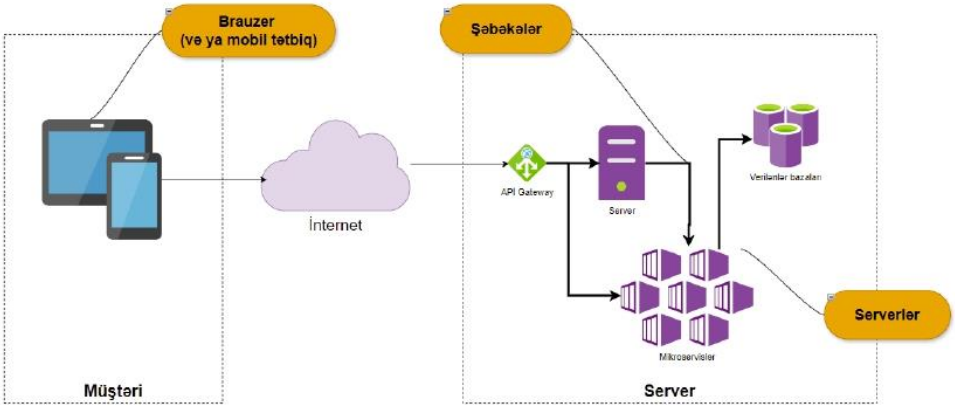
[2] MİDA-dan ev alanların NƏZƏRİNƏ: Martdakı nəticələr ləğv edildi (https://oxu.az/society/621103)

[3] Azərbaycanın dövlət strukturlarına haker hücumları edilib (https://az.trend.az/azerbaijan/society/2963683.html).

[4] XRİTDX tərəfindən 373 dövlət informasiya sistemində təhlükəsizlik boşluqları və nöqsanları aşkar edilmişdir (https://scis.gov.az/az/news/view/134/xritdx-terefinden-373-dovlet-informasiya-sisteminde-tehlukesizlik-bosluqlari-ve-noqsanlari-askar-edilmisdir)

Veb təhlükəsizliyin tərkib hissələri

Veb təhlükəsizliyin əsas məqsədinin veb proqram təminatlarının fasiləsiz şəkildə işləməsini təmin etmək və həmin proqramlara qarşı yönəlmiş bir sıra hücumların qarşısını almaqdan ibarət olduğunu qeyd etmişdik. Hal-hazırda istifadə etdiyimiz bir çox veb sayt və ya mobil tətbiq əslində yalnız interfeys rolunu oynayır. Onların arxasında nəhəng bir infrastruktur ola bilər. Bir çox veb tətbiqlərin arxasında aşağıdakı şəkildəki kimi server infrastrukturu dayanır. Təbii ki, bu infrastruktur adətən daha mürəkkəb və detallı olur, verilən şəkil bu ekosistemin ümumi təsviridir.



Şəkil 1.5: Veb tətbiqlər və onlara xidmət edən server infrastrukturuna nümunə

Beləliklə, veb təhlükəsizlik dedikdə iştirak edən bütün sistemlərin təhlükəsizliyi nəzərdə tutulur. Bu kursun kontekstində veb təhlükəsizliyi aşağıdakı üç qrupa bölməcəyik:

- Brauzer təhlükəsizliyi
- Server təhlükəsizliyi
- Şəbəkə təhlükəsizliyi

Brauzer təhlükəsizliyi

Veb brauzer – veb-səhifələri göstərmək və naviqasiya etmək üçün qrafik istifadəçi interfeysi olan kompüter proqramıdır. Əminəm ki, siz hər gün veb brauzerdən istifadə edirsiniz. Hazırda Google Chrome, Microsoft Edge, Firefox, Safari, Brave, Samsung İnternet və s. kimi məşhur veb brauzerlər geniş yayılmışdır. Brauzerlərin necə işləməsinə dair daha ətraflı dördüncü mühazirədə öyrənəcəyik. Brauzer təhlükəsizliyi dedikdə istifadəçiləri internetdən brauzer vasitəsilə istifadə edərkən onlayn təhlükələrdən qorunması başa düşülür. Brauzerlər istifadəçilər və internet arasında əsas pəncərə kimi tez-tez kiberhücumların hədəfi olurlar. Elə veb tətbiqlər də əslində daha dar çərçivədə istifadəçiləri həmin tətbiqə aid xidmətlərə çıxışını təmin edir. Bu baxımdan brauzer təhlükəsizliyi dedikdə

həmin tətbiqlərin təhlükəsizliyi də başa düşülür.

Brauzerlərə hədəflənmiş Kiberhücumların aşağıdakı növlərini sadalamaq olar:
Fişinq Hücumları: Qanuni veb saytları təqlid edərək istifadəçiləri aldadan şəxsi və ya maliyyə məlumatlarını oğurlamaq məqsədli aldadıcı üsullar.
Zərərli proqram: İstifadəçinin sistemindəki məlumatlara zərər vermək, istismar etmək və ya oğurlamaq üçün nəzərdə tutulmuş zərərli proqram. Zərərli yükləmələr və ya təhlükəsi olan veb saytlar vasitəsilə çatdırıla bilər. Bəzi zərərli proqramlar brauzer proqramlarındakı boşluqları istifadə edərək sizdən xəbərsiz kompüterinizi və ya mobil qurğunuzu yoluxdura bilər.
Saytlararası Skriptləmə (XSS): Hakerlər tərəfindən zərərli skriptləri etibarlı vebsaytların məzmununa daxil etməyə imkan verən boşluq. Bu boşluqdan istifadə edərək qanuni saytlardan sizin məlumatlarınızı əldə etmək və ya sizin adınızdan əməliyyatlar yerinə yetirmək mümkün ola bilər.
Saytlararası Tələb Saxtakarlığı (CSRF): İstifadəçini hazırda autentifikasiya olunduğu veb proqramda arzuolunmaz hərəkətləri yerinə yetirməyə məcbur edən hücum.
Man-in-the-Middle (MitM) Hücumları: Təcavüzkar mübadilə edilən məlumatları oğurlamaq və ya manipulyasiya etmək üçün iki tərəf arasındakı əlaqəni kəsdikdə və ya araya daxil olaraq məlumatları əldə etdikdə baş verən hücumlar.

Biz bu mövzularla növbəti mühazirələrdə daha ətraflı tanış olacağıq. Brauzer təhlükəsizliyi rəqəmsal dünyada təhlükəsiz naviqasiyanın vacib elementidir. Ümumi təhdidləri başa düşmək və ən yaxşı təcrübələrə riayət etməklə istifadəçilər kiberhücumların qurbanı olmaq riskini əhəmiyyətli dərəcədə azalda bilərlər. Kibertəhlükəsizlik təhdidləri inkişaf etməyə davam etdikcə, onlardan müdafiə strategiyaları da bütün internet istifadəçiləri üçün davamlı maarifləndirmə mühüm əhəmiyyət kəsb edir.

Server təhlükəsizliyi

Server təhlükəsizliyi məlumatları və proqram təminatlarını icazəsiz girişdən və digər kiber təhlükələrdən qorumaq üçün istifadə olunan prosesləri, alətləri və siyasətləri əhatə edir. Veb serverlər, e-poçt serverləri, verilənlər bazası serverləri və ya hər hansı digər növ serverlər təşkilatın İT infrastrukturunun kritik komponentləridir, çox vaxt həssas məlumatları saxlayır və istifadəçilərə və tətbiqlərə əsas xidmətlər təqdim edir. Server dedikdə ənənəvi olaraq avadanlıq təminatı deyil, həm də virtual maşınları, konteynerləri, onlar əsasında çalışan mikroservisləri də nəzərdə tuturuq.

Serverlər saxladıqları dəyərli məlumatlara və təqdim etdikləri xidmətlərə görə hakerlər üçün əsas hədəfdir. Hər hansı bir haker hücumu məlumatların oğurlanmasına, müştəri etibarının itirilməsinə, maliyyə zərərinə və hüquqi fəsadlara səbəb ola bilər. Buna görə də, serverlərin təhlükəsizliyini təmin etmək təkcə məlumatların qorunması deyil, həm də təşkilatın nüfuzunun qorunması və qanunvericiliyə əməl olunması məsələsidir.

Server təhlükəsizliyi mürəkkəb və davamlı prosesdir. Müasir zamanda bir virtual maşın-

da və ya mikroservisin çalışdığı konteynerdə yüzlərlə bəzən minlərlə proqram təminatı, onların istifadə etdikləri kitabxanalar istifadə olunur. Hakerlər davamlı olaraq həmin proqramlarda aşkar etdikləri zəifliklərdən istifadə edirlər.

Server təhlükəsizliyinin əsaslarını başa düşmək İT resurslarının idarə edilməsində iştirak edən hər kəs üçün çox vacibdir. Güclü təhlükəsizlik tədbirləri həyata keçirməklə və təhlükəsizliyə proaktiv yanaşma tətbiq etməklə təşkilatlar kiberhücumlara qarşı həssaslığını əhəmiyyətli dərəcədə azalda, məlumat və xidmətlərini qoruya bilər.

Şəbəkə təhlükəsizliyi

Bu gün şəbəkəyə qoşulmamış informasiya sistemi təsəvvür etmək çətindir. İstər mobil telefonlar, istər dövlət qurumlarındakı sistemlər, veb saytlar, banklar və təhsil ocaqları, hər biri şəbəkələrdən ibarət sistemlər çoxluğundan istifadə edir. Onların bir çoxu hətta internet vasitəsilə hansısa formada bir-birilə əlaqələrə malikdir. Hazırda yer kürəsini böyük bir kompüter şəbəkə toru əhatə edir.

Şəkil 1.6: Hazırda yer kürəsini böyük bir kompüter şəbəkə toru əhatə edir.

Yuxarıda brauzerlər və serverlər haqqında, onların təhlükəsizliyi barədə söhbət açdıq. Onları bir-birilə əlaqələndirən vasitə şəbəkədir. Ona görə də şəbəkələrin nə olduğunu bilmək və şəbəkələrin təhlükəsizliyini anlamaq veb təhlükəsizlik üçün olduqca vacibdir.

Kompüter şəbəkələri müxtəlif şəbəkə qurğuları, kabellər, işıq şüaları və radio dalğalar vasitəsilə qurulur. Həmin qurğular özləri də əslində kompüterlərdir, onların da əməliyyat sistemləri var. Bu əməliyyat sistemləri də müxtəlif proqram təminatlarına malikdir. Bu o deməkdir ki, serverlərdə olduğu kimi, şəbəkə avadanlıqlarında da təhlükəsizlik boşluqları

ola bilər. Digər məqam da ondan ibarətdir ki, şəbəkə qurğuları bəzən bizim nəzarətimizdə olmayan məkanlarda ola bilər. Məsələn, biz çalışdığımız idarənin başqa şəhərdə olan bir şöbəsi ilə şəbəkə əlaqəsi qurmaq üçün hansısa internet provayderdən və ya kiməsə aid olan şəbəkə xidmətindən istifadə edirik. Bəs onun təhlükəsizliyinə necə nəzarət etmək olar?

Şəbəkə təhlükəsizliyi həm proqram təminatı, həm də aparat texnologiyalarından istifadə edərək kompüter şəbəkələrinin və məlumatların tamlığını, konfidensiallığını və əlçatanlığını qorumaq üçün qoruyucu tədbirlərin və protokolların həyata keçirilməsi təcrübəsidir. Ölçüsündən, sənayesindən və ya infrastrukturundan asılı olmayaraq hər bir təşkilat onu rəqəmsal dövrdə artan kiber təhlükələrdən qorumaq üçün müəyyən dərəcədə şəbəkə təhlükəsizliyi həlləri tələb edir. Şəbəkə təhlükəsizliyinin əsas məqsədi təşkilat aktivlərini çoxsaylı təhdidlərdən qorumaq və onların yayılmasının və böyüməsinin qarşısını almaqdır. Bu təhdidlərə casus proqramlar, fişinq, ransomware və s. daxil ola bilər. Effektiv şəbəkə təhlükəsizliyi şəbəkəyə girişi idarə edərək, düzgün etimadnaməyə malik olanların ehtiyac duyduqları resurslara daxil ola bilməsini, olmayanların isə kənarda qalmasını təmin edir. Bir çox hallarda təhdidlər heç də zərərverici proqramlar vasitəsilə deyil, səhv konfiqurasiya ucbatından da baş verir.

Şəbəkə təhlükəsizliyində əsas problem kibertəhlükələrin daimi təkamülü ilə ayaqlaşmaqdır. Hakerlər davamlı olaraq yeni zəifliklər tapırlar və ona görə də təhlükəsizlik tədbirlərinin oxşar sürətlə inkişaf etməsini vacibdir. Bundan əlavə, şəbəkələr gün-gündən daha da mürəkkəbləşir. Mürəkkəb şəbəkələri başa düşmək, daim nəzarətdə saxlamaq olduqca çətin işdir.

Şəbəkə təhlükəsizliyi hər hansı bir təşkilat üçün İT-nin vacib aspektidir. Effektiv şəbəkə təhlükəsizliyi təşkilatın aktivlərini bugünkü rəqəmsal dünyada üzləşdiyi saysız-hesabsız təhlükələrdən və təşkilatın nüfuzuna xələl gəlməkdən qoruyur. Şəbəkə təhlükəsizliyinin əsas elementlərini başa düşmək və həyata keçirməklə təşkilatlar kibertəhlükəsizlik riskini əhəmiyyətli dərəcədə azalda bilər.

Veb təhlükəsizlik kursunun strukturu və gözlənilən nəticələr

Kursun strukturu

Kurs haqqında əhatəli təsəvvürə malik olmaq, onu düzgün qavramaq üçün kursda keçiləcək mövzular barədə əvvəlcədən xəbərdar olmaq lazımdır. Ona görə də kursun strukturu barədə bu bölmədə məlumat vermək istərdik.

Veb təhlükəsizlik kursunun ilk bölməsi tələbələrə kurs haqqında ümumi məlumatları, kursdan gözlənilən nəticələr barədə xəbər verir.

Kursun ikinci və üçüncü mühazirələrində ümumiyyətlə internetin nə olduğunu, brauzerin veb səhifələri necə oxuması, DNS sisteminin necə çalışması, HTTP protokolu və şəbəkə ilə bağlı ilkin məlumatları izah edir.

Veb təhlükəsizliyi təmin etmək üçün əlbəttə ki, veb texnologiyaları, veb səhifələrin hazırlanmasını, JavaScript dilini bilmək vacibdir. Ona görə də kursun dördüncü və beşinci mühazirələri HTML işarələmə dili, CSS, JavaScript dili və onlara aid bəzi texnologiyalardan bəhs edir.

Altıncı mühazirə tələbələrə bəzi alətləri – brauzerdə DevTools, sistemdə olan curl, nslookup, tracert və s. kimi alətləri istifadə etməyi öyrədir. Həmçinin snifferlər barədə izah edir və Wireshark proqram təminatı ilə şəbəkə trafikinə nəzarət etmək izah olunur.

Veb təhlükəsizlikdə bir sıra hücumlar istifadəçi sessiyalarının ələ keçirilməsi ilə həyata keçirilir. Ona görə də sessiyaların necə çalışdığını öyrənmək olduqca vacibdir. Yeddinci mühazirə tamamilə istifadəçi sessiyaları, brauzer və server arasında sessiyalar haqqında məlumatların mübadiləsi və əlaqədar mövzular haqqındadır.

Səkkizinci mühazirədə kod inyeksiyası barədə ilkin məlumat verilir və XSS haqqında öyrədilir. Mühazirə təcrübi tapşırıqlarla davam edir.

Doqquzuncu mühazirə xidmətdən imtina (DoS) hücumlarına həsr olunmuşdur.

Onuncu mühazirədə fişinq hücumları izah edilir. Təcrübə hissəsində fişinq hücumunun həyata keçirilməsi izah edilir.

On birinci və on ikinci mühazirələr SQL dili, nümunə olaraq MySQL verilənlər bazası və SQL inyeksiyası haqqında öyrədir.

On üçüncü mühazirədə kriptoqrafiya haqqında giriş məlumatı verilir, simmetrik və assimmetrik şifrələmə, heş funksiyalar barədə məlumat verilir, onların veb təhlükəsizlikdə rolu, proqram təminatlarında istifadə üsulları haqqında danışılır. Mühazirə kriptoqrafik üsulların köməyi ilə SSL protokolunun izahı ilə davam edir.

On dördüncü mühazirədə autentifikasiya, avtorizasiya, şifrələrin etibarlılığı barədə söhbət açılır.

Nəhayət sonuncu mühazirədə veb təhlükəsizliklə bağlı müasir problemlər barədə diskussiya yer alması nəzərdə tutulmuşdur. Burada istifadəçilərin izlənməsi, süni intellekt, blokçeyn texnologiyaları barədə mövzular yer ala bilər və onların veb təhlükəsizlik məsələsində yeri barədə danışıla bilər. Həmin mövzular həddindən artıq yüksək sürətlə yenilənir. Biz bu mühazirələr toplusunu hazırlayarkən on beşinci mühazirə üçün nəzərdə tutduğumuz mövzuların bir hissəsi artıq aktuallığını itirmiş və yeni texnologiyalar əmələ gəlmişdi. Çaşqınlıq yaranmaması və praktiki əhəmiyyəti olmayan, aktuallığını itirmiş mövzuları əks etdirməmək üçün on beşinci mühazirəni bu fənni tədris edən müəllimin ixtiyarına buraxmağa qərar verdik.

“Veb təhlükəsizlik” fənnindən gözlənilən təlim nəticələri

Hər bir kursun vəzifəsi ölkə üçün kadr potensialının yüksəldilməsi, güclü mütəxəssislərin yetişdirilməsində rol oynamaqdır. Kibertəhlükəsizlik mütəxəssislərinin hazırlıq prosesi-

nin müasir standartlara cavab verməsi üçün təhsil proqramları daim yenilənir. Bu mühazirə hazırlanan zaman, bakalavriat səviyyəsinin "İnformasiya təhlükəsizliyi" ixtisası üzrə təhsil proqramı 2022-ci ildə qəbul edilmiş proqram olmuşdur[5].

Həmin təhsil proqramına görə "Veb təhlükəsizlik" fənnindən gözlənilən təlim nəticələri aşağıdakılardır:

FTN 1 - Veb brauzerlərin iş prinsiplərini, veb tətbiqlərin layihələndirilməsi, yaradılması və istismarı qaydalarını bilməlidir.

FTN 2 - Müasir texnologiyalardan istifadə edərək təhlükəsiz veb səhifələr hazırlamağı bacarır. HTML, CSS, JavaScript kimi texnologiyaları bilməlidir.

FTN 3 - HTTP protokolunun necə çalışdığını bilir. SSL sertifikatlarının necə çalışdığını və veb səhifələrin təhlükəsizliyini necə qoruduğunu bilir.

FTN 4 - Veb saytlarda hücumların təbiəti haqqında məlumatlıdır.

FTN 5 - REST və GraphQL kimi veb xidmətlərin necə çalışdığını bilir.

FTN 6 - OWASP Top 10 - təhlükəsizlik zəifliklərı ilə tanışdır, o siyahıları müəyyənləşdirməyi, onların qarşısının alınması mexanizmlərini bacarır.

Kursun məzmunu həmin bu 6 FTN əsasında qurulmuşdur.

Təcrübə

Kibertəhlükəsizlik mütəxəssisləri öz fəaliyyətləri üçün lazım olan istər proqram, istərsə də avadanlıq vasitələrini istifadə etməyi bacarmalıdırlar. Bu mütəxəssislər kiber-cinayətkarları yaxşı anlamalıdır, onları başa düşməli, onlar kimi düşünə bilməlidir. Öyrənmə və sınaq məqsədilə özünü kiber-cinayətkar yerinə qoymalı, zəifliklərı tapmalı və sonra bu bilikləri istifadə edərək müdafiə mexanizmlərini yaratmalıdırlar. İstifadə etdikləri alətlər hər zaman əlçatan olmalıdır.

Veb təhlükəsizlikdə siz bir çox alətləri öyrənəcək və dərs boyu onlardan istifadə edəcəksiniz. Növbəti paraqraflarda həmin alətlərin bəziləri ilə tanış olacaq, onların kompüterinizdə yüklənməsini öyrənəcəksiniz.

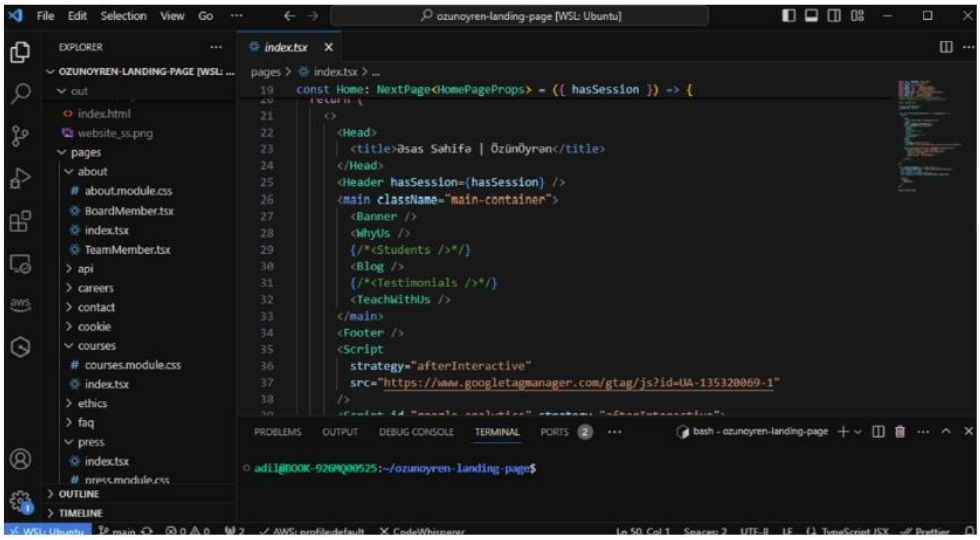
Visual Studio Code

İnformasiya texnologiyalarının hansı sahəsi ilə məşğul olmanızdan asılı olmayaraq tez-tez fərqli mətn fayllarını redaktə etmək və ya oxumaq istəyəcəksiniz. Bu fayllar sənədlər, konfiqurasiya faylları, proqram mətnləri, sistem skriptləri, elektron imzalar və s. ola bilər. Mətnləri işləmək üçün sizə mətn redaktoru lazım olacaq. Bir çox əməliyyat sistemlərinin tərkibində sadə mətn redaktoru mövcud olur. Lakin, bu redaktorların imkanları məhdud-

[5] Azərbaycan Respublikası Təhsil Nazirinin 28 İyul 2022-ci il tarixli, F-463 saylı əmri ilə təsdiqlənmiş, Bakalavriat səviyyəsinin "İnformasiya təhlükəsizliyi" ixtisası üzrə təhsil proqramı

dur.

Visual Studio Code və ya qısa olaraq VSCode bir çox proqramçının sevdiyi mətn redaktorudur. VSCode vasitəsilə siz fərqli proqramlaşdırma dillərində yazılmış kodları oxuya, yaza bilərsiniz. Proqram mətninin sintaksisinə görə fərqli rənglərdən istifadə etmək, mətnin strukturunu avtomatik formatlama etmək mümkündür. VSCode-un fərqli genişlənmələri ilə daha çox əməliyyatlar yerinə yetirmək mümkündür.



Şəkil 1.7: Visual Studio Code proqramı

HTML, JavaScript, CSS, SQL və digər kodları yazmaq üçün VS Code proqramını geniş istifadə edəcəksiniz.

VS Code-u yükləmək üçün https://code.visualstudio.com/ saytına baxın. Saytda onu yükləmək üçün təlimatlar yazılmışdır.

Ubuntu for Windows

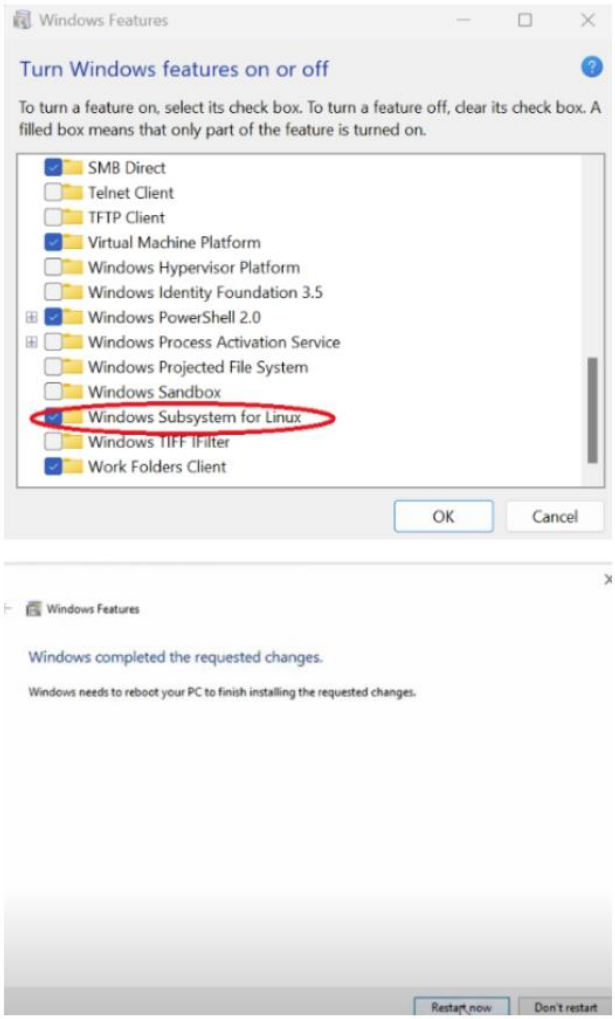
Kursda istifadə edəcəyiniz alətlərin demək olar hamısı Linux əsaslı əməliyyat sistemlərinə aiddir. Əgər Windows əməliyyat sistemi işlədirsinizsə, onda sizə Ubuntu for Windows proqramı mütləq lazım olacaq. Ubuntu for Windows istifadə etməklə bir növ virtual maşının daxilində Ubuntu sistemi işə düşəcək. Ubuntu ən geniş istifadə olunan Linux distributivlərindən biridir.

Ubuntu for Windows yükləmək üçün ardıcıllıq:

1. Control Panel bölməsində Programs/Programs and Features/ seçilir
2. Pəncərinin sol hissəsində yerləşən Turn Windows features on or off bölməsi seçilir.
3. Açılan yeni kiçik pəcərədə Windows Subsystem for Linux hissəsi aktiv edilir. Və

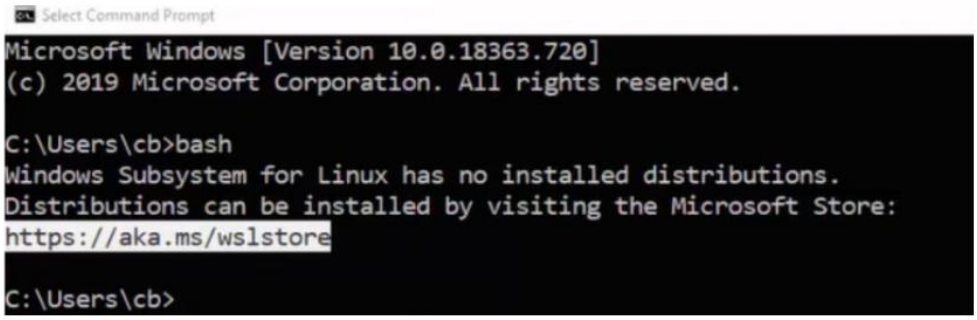
komputer yenidən başladılır.

Şəkil 1.8: Windows ƏS-də “Programs and Features” bölməsi



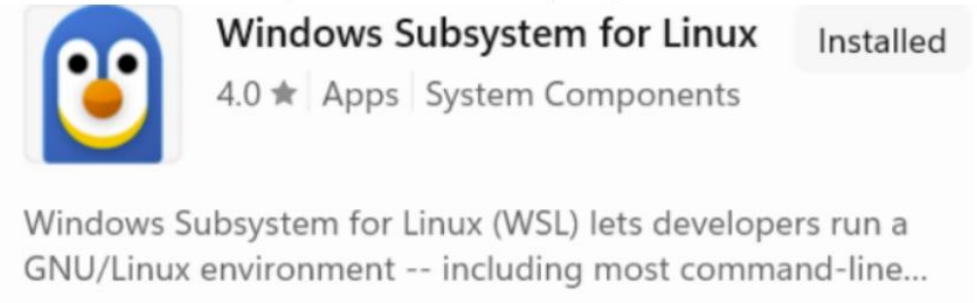
Şəkil 1.9: “Windows Subsystem for Linux” sisteminin aktivləşdirilməsi

5. Sonra “Command Prompt” adlanan Əmr Lövhəsi açılır bash əmrini yazaraq Windows Subsystem for Linux yüklü olub olmadığını yoxlayaq.



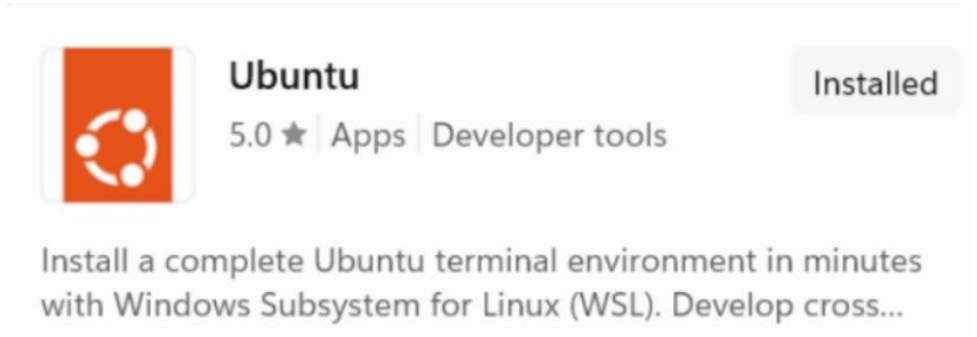
Şəkil 1.10: "Windows Subsystem for Linux"-un mövcudluğunun yoxlanılması

Şəkildə də göründüyü kimi yüklü olmadığını bildirir. Və yükləmək üçün bizə link təklif edir. Amma sadə olması üçün Microsoft Store-dən yükləyə bilərik.



Şəkil 1.11: "Windows Subsystem for Linux"-in Microsoft Store vasitəsilə yüklənməsi

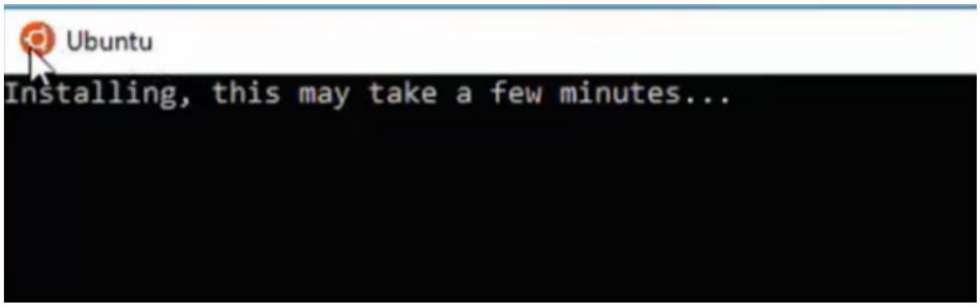
Bundan əlavə yükləməli olduğumuz ubuntu sistemi də var.



Şəkil 1.12: "Ubuntu for Windows"-un Microsoft Store vasitəsilə yüklənməsi

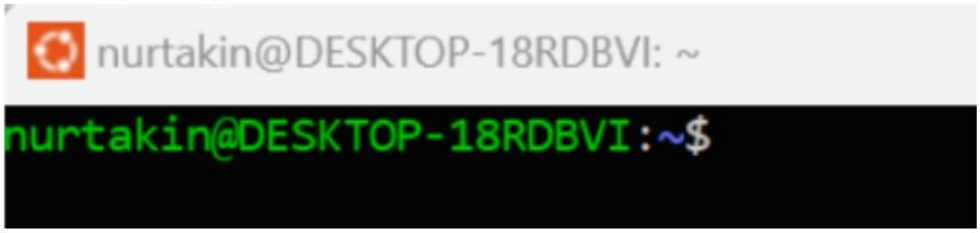
6. Yükləmə bitdikdən sonra, Windowsda "Start" menyusu vasitəsilə Ubuntu axtarılır.

Və budur artıq Ubuntu terminalı!



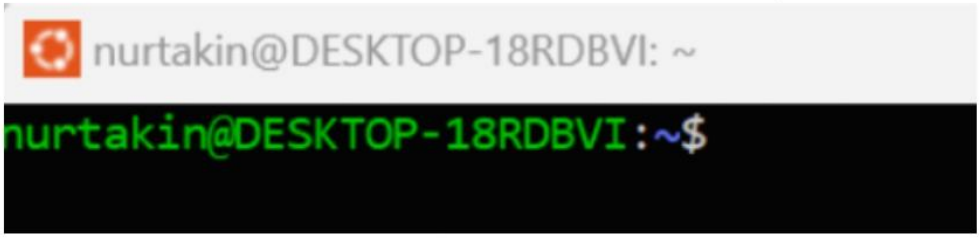
Şəkil 1.13: “Ubuntu for Windows”un ilk dəfə işə salınması

7. Konfiqurasiya hissəsində bir az gönlədikdən sonra sizdən ad, parol və parolu yenidən daxil etməyinizi istəyəcək.



Şəkil 1.14: “Ubuntu for Windows”un ilk dəfə işə salınması zamanı şifrənin təyin edilməsi

Bütün bu proseslər bitdikdən sonra artıq Ubuntu terminalımız yüklənmiş olacaq.



Şəkil 1.15: “Ubuntu for Windows”un ilk dəfə işə salınması

Yüklənmə bitdikdən sonra belə bir yazı ilə qarşılacaqsınız. Bu o deməkdir ki, artıq terminal işlək vəziyyətdərdir. İlk hissədə də gördüyünüz isə bir az əvvəl terminala daxil etdiyiniz UNİX username (adı) olacaq.

Kali linux

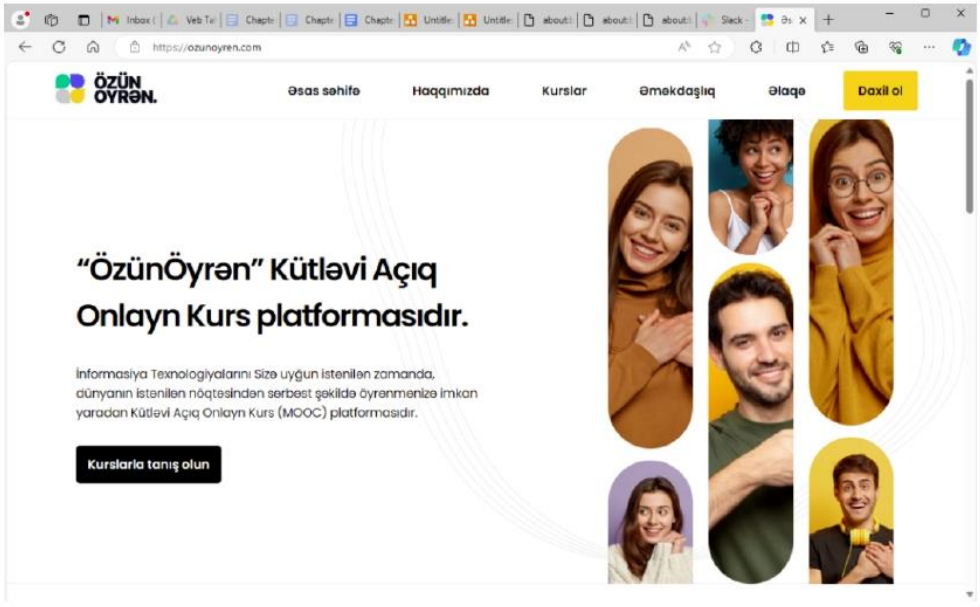
Əgər Linux istifadə etmək istəyirsinizsə təhlükəsizliyi öyrənmək üçün Kali Linux (https://www.kali.org/) istifadə etmək yerinə düşərdi. Bu distributivin içində kiber-təhlükəsizlik ilə əlaqədar bir çox maraqlı alətlər standart paketlər siyahısına daxil edilmişdir.

Kali distributivini yükləməklə həmin alətlərin hamısı əlinizin altında olacaq.

Brauzer (Firefox/Chrome/Edge)

Tədris boyu biz Firefox, Chrome, Brave və ya Edge brauzerini istifadə edəcəyik. Bu brauzerlərin DevTools funksiyası bir sıra təcrübələr üçün bizim köməyimizə çatacaqdır.

Əgər bu proqramlardan heç biri yoxdursa istənilən birini internetdən yükləyə bilərsiniz.



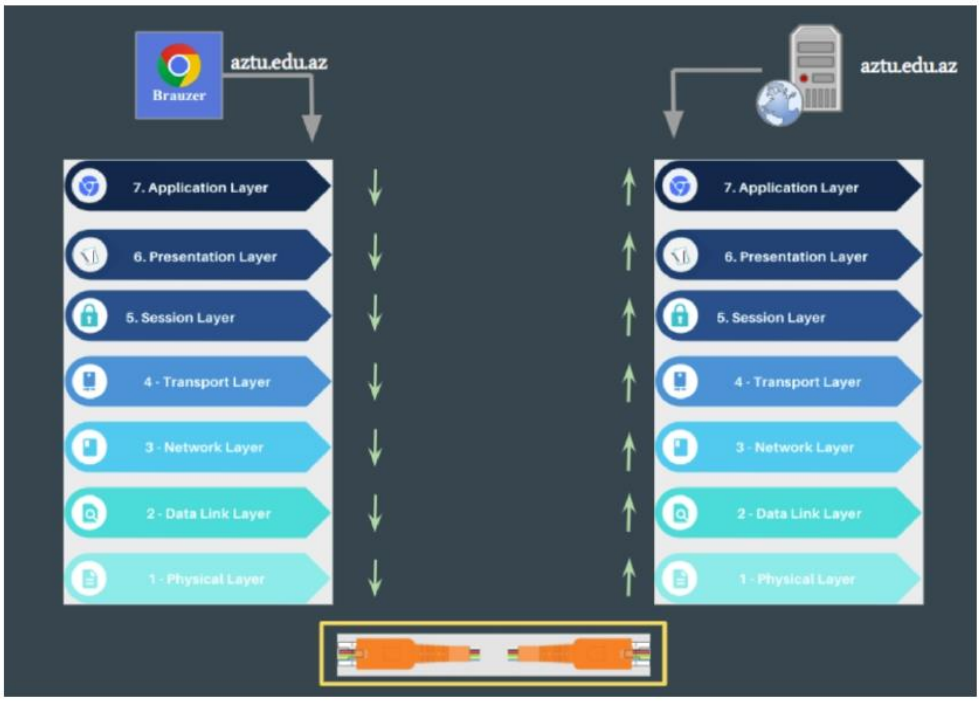
Şəkil 1.16: Microsoft Edge brazueri

Əgər Mac OS istifadə edirsinizsə sadaladığımız brauzerlərdən birini yükləyin. Mac OS sisteminin üzərində gələn Safari brauzeri bu kursun tapşırıqları üçün əlverişli deyildir.

2 | Mühazirə 2

Protokollar

Kompüter şəbəkələrində məlumatların ötürülməsi üçün məlumatlar proqram təminatında yaradılır, daha sonra bir neçə addımda elektromaqnit siqnalına çevrilərək naqillərlə ötürülür. Ünvana çatarkən həmin elektromagnit siqnalları rəqəmsal məlumata çevrilir. Bu addımlar 7 laydan ibarət OSİ şəbəkə laylarından keçir.



Şəkil 2.1: OSI modeli və OSI layları ilə məlumatın hərəkəti

OSİ laylarının hər-birinin öz vəzifəsi vardır və o layların hər biri müəyyən protokollar üzrə çalışır.

İnternet protokolları kompüterlərə internet və digər şəbəkələr üzərindən ünsiyyət qurmağa imkan verən qaydalar və ya standartlar toplusudur. Bu protokollar məlumat paketlərinin müxtəlif cihazlar arasında düzgün şəkildə göndərilməsini və qəbul edilməsini təmin edir. Əsas internet protokollarından bəzilərinə aşağıdakılar daxildir:

- TCP/IP
- HTTP və HTTPS
- DNS

Gəlin hər biri ilə ayrı-ayrılıqda tanış olaq. Adı çəkilən protokollar haqqında daha geniş

məlumat “Kompüter Şəbəkələri” və “Şəbəkə təhlükəsizliyi” fənnlərində əhatə edilir. Bu fənndə onlar haqqında ilkin məlumatlara toxunacağıq.

TCP/IP

Transmission Control Protocol/Internet Protocol mənasını verən TCP/IP modeli şəbəkələr, xüsusən də internet üzərindən rəqəmsal rabitə üçün təməl protokollar paketidir. TCP və İP ayrı-ayrı protokollardır, və şəbəkə rabitəsi üçün təməl protokollar olduğu üçün onların adı hər zaman birgə çəkilir. Bəzən yalnışlıqla ona TCP/IP protokolu da deyirlər. Lakin, əslində onlar protokollar çoxluğudur. TCP/İP protokollar paketi müxtəlif şəbəkələr üzrə məlumatların daşınmasını və yönləndirilməsini tənzimləyən, aparat, proqram təminatı və ya şəbəkə infrastrukturundan asılı olmayaraq cihazlar arasında etibarlı əlaqəni təmin edən bir sıra protokolları əhatə edir. Əslində, TCP/IP internetin əsasını təşkil edir.

Qeyd etdiyimiz kimi TCP/IP iki əsas protokolu birləşdirir: TCP və IP. IP, cihazlar arasında məlumat paketlərinin ünvanlanması və yönləndirilməsi üçün cavabdehdir, onların birbirinə bağlı şəbəkələr arasında göndərilməsinə və qəbul edilməsinə imkan verir. O, məlumat paketlərinin düzgün təyinat yerinə çatmasını təmin edərək, şəbəkədəki hər bir cihaza unikal IP ünvanları təyin etməklə işləyir.

IP protokolu məlumat paketlərini düzgün yerə yönləndirdikdən sonra məlumat ötürülməsinə cavabdehlik TCP protokolunun üzərinə düşür. İnternet üzərindən göndərilən məlumatların etibarlı və xətasız çatmasını təmin edir. TCP, göndərən və qəbuledən arasında əlaqə yaratmaq, səmərəli ötürülmə üçün böyük məlumat fayllarını daha kiçik paketlərə bölmək və sonra onları təyinat yerində yenidən yığmaq yolu ilə işləyir. O, həmçinin tranzit zamanı heç bir məlumatın itirilmədiyini və zədələnməməsini və paketlərin göndərildikləri ardıcıllıqla qəbul edilməsini təmin etmək üçün axına nəzarəti və xətaların yoxlanılmasını idarə edir.

TCP/IP protokollar paketi ilə cihazlar öz aralarında IP ünvanlar vasitəsilə əlaqə qururlar.

İP ünvan şəbəkəyə qoşulmuş avadanlıqlar arasında əlaqə üçün istifadə olunan unikal ünvandır. Hazırda geniş istifadə edilən İP ünvanlar “.” (nöqtə) işarəsi ilə ayrılmış 4 ədəddən ibarətdir. Bu ədədlərin ala biləcəyi qiymət 0-255 aralığındadır. Bu ünvanlar İPv4 adlanır. İnternetə qoşulmuş avadanlıqların sayı sürətlə artdığından onların sayı tükənmək üzrədir. Ona görə də IPv6 icad edilmişdir. IPv6 ünvanları 128 bit uzunluğundadır və mümkün ünvanların sayını əhəmiyyətli dərəcədə artırır. IPv6 ünvanı dörd onaltılıq rəqəmdən ibarət səkkiz qrup şəklində təmsil olunur, məsələn, 2001:0db8:85a3:0000:0000:8a2e:0370:7334.

IP ünvanları statik və ya dinamik ola bilər. Statik IP ünvanlar dəyişmir. Onlar idarəçi tərəfindən kompüterə əl ilə təyin edilir və zamanla sabit qalır. Statik IP ünvanları tez-tez veb saytları yerləşdirən serverlər üçün istifadə olunur, çünki domen adının bu ünvanlara pərçimlənməsi üçün daimi ünvan lazımdır.

Dinamik IP ünvanları cihazlar hər dəfə şəbəkəyə qoşulduqda müvəqqəti təyin edilir. Di-

namik IP ünvanları adətən mövcud nömrələr çoxluğundan IP ünvanlarını avtomatik təyin edən DHCP tərəfindən təyin olunur. Əksər ev şəbəkələri dinamik IP ünvanından istifadə edir.

HTTP və HTTPS

HTTP (Hypertext Transfer Protocol) və HTTPS (HTTP Secure) internetdə məlumat ötürmək üçün istifadə olunan əsas protokollardır, ilk növbədə mesajların necə formatlaşdırıldığını və ötürüldüyünü və veb serverlərin və brauzerlərin müxtəlif əmrlərə cavab olaraq hansı tədbirləri görməli olduğunu müəyyən edir.

HTTP protokolu OSİ modelinin "Application" layına aiddir. HTTP müştərilər və serverlər arasında əlaqə yaratmaq üçün nəzərdə tutulmuşdur. İstifadə etdiyiniz brauzerlər məlumatlar mübadiləsini server ilə HTTP protokolu vasitəsilə yerinə yetirir. Veb-brauzer serverə HTTP sorğusu göndərir, sonra isə serverdən cavab alır. Cavab sorğu haqqında status məlumatını ehtiva edir və həmçinin veb səhifə, şəkil və ya video kimi tələb olunan məzmunu ehtiva edə bilər. HTTP ilə məlumatların ötürülməsi zamanı müxtəlif metodlar – GET, POST, DELETE, PUT və s. istifadə edilir. Bu metodlar barədə növbəti dərslərimizdə daha ətraflı öyrənəcəyik.

HTTP protokolu ilə məlumatlar açıq şəkildə ötürülür. Yəni məlumat ötürülərkən kimsə onu yolda əldə edə bilsə onu oxuya biləcək. Bu isə böyük təhlükəsizlik riski deməkdir. Məsələn, siz HTTP protokolu ilə çalışan veb sayt üzərində alış-veriş edərkən, kredit kart nömərinizi daxil etdikdə onu kimsə əldə edə bilər. Bu problemi həll etmək üçün məlumat müştəri ilə server arasında şifrələnmiş formada ötürülməlidir. Bu ideyanın əsasında HTTP-nin başqa bir versiyası HTTPS yaradılmışdır. HTTPS HTTP-nin təhlükəsiz versiyasıdır, burada "S" hərfi "Təhlükəsiz" (Secure) deməkdir. Bu o deməkdir ki, brauzeriniz və veb saytınız arasındakı bütün əlaqə şifrələnir. Növbəti dərslərimizdə HTTP və HTTPS-in detalları barədə ətraflı öyrənəcəyik.

DNS

DNS internetin digər vacib komponentidir və insan üçün rahat oxunan domen adlarını (məsələn, www.akta.az) IP ünvanlarına tərcümə edən kataloq kimi fəaliyyət göstərir. Brauzerinizə veb ünvanını daxil etdiyiniz zaman, DNS serverləri həmin domen adını götürür və onu müvafiq IP ünvanına tərcümə edərək brauzerinizin veb-saytın serverinə qoşulmasına şərait yaradır.

İnternet

İnternet nədir?

Geniş və mürəkkəb qlobal şəbəkə olan İnternet müasir həyatın ayrılmaz hissəsinə çevrilərək bizim ünsiyyət, işləmə və məlumat əldə etməyimizi üçün əsas vasitədir. İnternet müxtəlif elektron, simsiz və optik şəbəkə texnologiyaları vasitəsilə dünya üzrə milyon-

larla özəl, ictimai, akademik, biznes və hökumət şəbəkələrini birləşdirən şəbəkələr şəbəkəsidir. O, Ümumdünya Şəbəkəsi (WWW), e-poçt, telefoniya və fayl mübadiləsi kimi geniş çeşidli xidmətləri asanlaşdırır.

İnternetin özəyi İnternet Protokolu (IP) və TCP protokolu vasitəsilə çalışır. İP məlumat paketlərini mənbədən təyinat yerinə göndərmək, ən yaxşı yolu tapmaq üçündür. TCP etibarlı məlumat ötürülməsini təmin edir, düzgün paket çatdırılmasını yoxlayır. Adətən bu iki protokolun adı birgə TCP/IP protokollar paketi kimi çəkilir.

İnternetin icadı 1950-1960-cı illərdə başlayıb və 1960-cı illərin sonunda ARPANET-in yaradılması ilə geniş vüsət almağa başladı. ABŞ Müdafiə Nazirliyi tərəfindən maliyyələşdirilən ARPANET paket kommutasiyasından istifadə edən ilk şəbəkə idi. 1970-ci illərdə Vint Cerf və Bob Kahn tərəfindən TCP/IP-nin inkişafı İnternetin əsasını qoydu və ARPANET 1983-cü il yanvarın 1-də TCP/İP ilə çalışmağa başladı. Bu tarix bizim bu gün bildiyimiz İnternetin yaradılma günü hesab edilə bilər.

İsveçrənin CERN təşkilatında çalışan Tim Berners-Li 1989-cu ildə HTML, URL və HTTP anlayışlarını gətirərək İnterneti ictimaiyyət üçün əlçatan etdi. Belə ki, ilk dəfə SGML adlı işarələmə dilinin əsasında HTML dilini yaratdı və orada bir səhifədən digərinə keçid (hiperlink) ideyasını reallaşdırdı. Bununla CERN işçiləri üçün sənədləri bir veb serverdə yerləşdirməyə və onlar arasında keçid etməyə müvəffəq oldu. 1993-cü ildə veb-brauzerlərin, xüsusən Mosaic-in buraxılması İnternetin böyüməsini daha da asanlaşdırdı. Daha sonra İnernet Explorer, Netscape Navigator kimi brauzerlər geniş yayılmağa başladı.

Şəkil 2.2: Ümumdünya şəbəkə toru.

Bu gün internet böyük bir məlumat külliyatını və elektron xidmətləri özündə ehtiva edən nəhəng bir şəbəkədir. O yer kürəsini böyük bir hörümçək toru kimi əhatə edir deyə ona ümumdünya hörmçək toru da deyirlər. Elə www – World Wide Web kəliməsi də buradan qaynaqlanır. Biz hər gün dərs oxuyarkən, bizneslə məşğul olarkən, asudə vaxt keçirərkən, alış-veriş edərkən internetdən istifadə edirik.

Brauzerdə URL yazdıqdan sonra nə baş verir?

İnternetin nəhəng bir şəbəkə toru olduğunu qeyd etdik, bəs bu şəbəkə torunda necə "səyahət etmək" olur? Biz adətən brauzeri açıb hansısa veb saytın ünvanını yazırıq, və ya bilmədiyimiz ünvanı axtarmaq üçün Google, Bing kimi axtarış sistemlərindən istifadə edirik. Bəs həmin ünvanı brauzerin ünvanlar xanasına yazıb "Enter" basdıqdan sonra nə baş verir? Brauzer bizə lazım olan saytı internetdən necə gətirib bizim ekranımızda bir neçə saniyə ərzində göstərə bilir?

Şəbəkələrə qoşulmuş hər bir qurğunun İP ünvanı olur. İnternet özü də bir şəbəkə olduğu üçün ona qoşulmuş bütün avadanlıqların İP ünvanı var. Siz əgər aztu.edu.az saytına daxil olmaq istəyirsinizsə, o sayta xidmət edən serverin öz İP ünvanını öyrənmək lazımdır. Məsələn, həmin İP ünvan 85.132.79.236 ola bilər. İP ünvanları yadda saxlamaq çətindir. Digər tərəfdən isə onlar dəyişə də bilər (Bu barədə mühazirənin növbəti bölməsində ətraflı danışacağıq). Ona görə də daha rahat yadda qalan ünvanlardan – domen adlarından istifadə edilir. Məsələn, aztu.edu.az domen adıdır. Amma bu domen adını İP ünvana tərcümə edən bir vasitə lazımdır.

Brauzerinizə veb-sayt ünvanını (URL) daxil etdiyiniz zaman ilk addım DNS bağlantısı yaratmaqdır. DNS internetin telefon kitabçası rolunu oynayır, insanın yadda saxlaya biləcəyi domen adlarını (məsələn, www.aztu.edu.az) kompüterlərin şəbəkədə bir-birini tanımaq üçün istifadə etdiyi IP ünvanlarına (məsələn, 85.132.79.236) tərcümə edir.

Şəkil 2.3: DNS əsasında domen adına uyğun İP ünvanın əldə edilməsinin sxematik təsviri.

Yuxarıdakı şəkildə sxetmatik olaraq İP ünvanın əldə edilməsi yazılmışdır. Amma əslində proses şəkildəki qədər sadə deyil. Kompüterimiz domen adı ilə əlaqəli IP ünvanlarını bilmir. Bu ünvanı əldə etmək üçün DNS resolver-lərdən istifadə edilir. Şəkildə DNS Server adlı sxem həmin DNS resolveri sadələşmiş formada təsvir edib.

Gəlin, DNS resolver-in necə işlədiyini araşdıraq. İlk olaraq DNS resolver dünyada məhdud sayda olan "root server"lər adlanan xüsusi DNS serverlərə müraciət edir. Onların

siyahısı aşağıda verilmişdir.

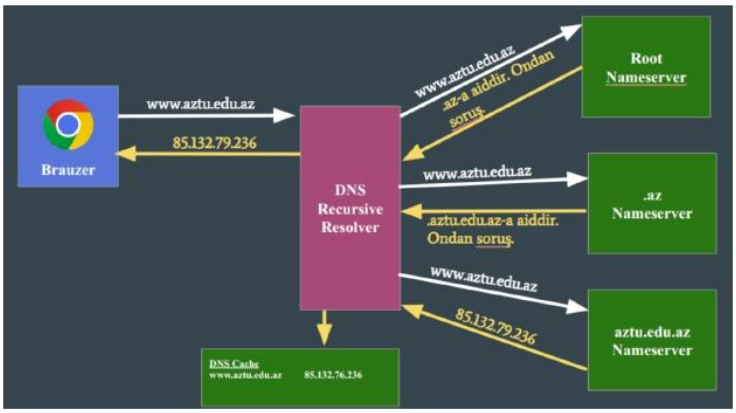
List of Root Servers

HOSTNAME	IP ADDRESSES	OPERATOR
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	170.247.170.2, 2801:1b8:10::b	University of Southern California, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

Şəkil 2.4: Root serverlərin siyahısı
(Mənbə: https://www.iana.org/domains/root/servers)

Root serverlərin siyahısını DNS resolverlər adətən İP ünvanlarına görə bilir. Bu ünvanlar dəyişmir (və ya nadir hallarda dəyişir). Ona görə də hər bir DNS resovlerin kiçik verilənlər bazasında bu siyahı olur. Root serverlər bütün domenlər barədə məlumatları özündə saxlamır. Onlar yalnız yüksək səviyyəli domenlərin (TLD – top level domain) "sahib"lərini tanıyır.

DNS resolver "root server"dən www.aztu.edu.az soruşanda o bildirir ki, onda belə bir ünvanın İP ünvanı mövcud deyil. Amma o domenin sonluğuna görə onun sahibinin ünvanını verir. Bu .az domen serveridir. Bu serverdə .az ilə bitən bütün müştərilərin məlumatları var. O isə deyir, mən bu domenin aztu.edu.az-a aid olduğunu bilirəm və mənim bazamda aztu.edu.az-ın İP ünvanı budur. Daha sonra DNS resolver daha aşağı səviyyəli domen serverindən İP ünvanı soruşmaq istəyir ki, bu da aztu.edu.az domenidir. Nəhayət DNS resolver aztu.edu.az serverindən "www.aztu.edu.az" domeninə aid İP ünvanı soruşur və əldə edir.



Şəkil 2.5: DNS resolverin iş prinsipi.

Bu proses kifayət qədər uzun prosesdir. Və hər dəfə bu serverləri sorğulamaq vaxt alır. Ona görə də məlumat əldə edildikdə o müvəqqəti yaddaşda – keş yaddaşında qalır. Gələn dəfə kimsə həmin DNS resolverdən www.aztu.edu.az domeninin İP ünvanını soruşanda o həmin sorğulamaları bir daha etmir və dərhal öz bazasından təqdim edir. Lakin, həqiqətdə İP ünvan dəyişə bilər. Ona görə keş yaddaşda məlumatlar daimi qalmamalıdır. Bəs nə qədər müddət qalmalıdır. Bunun üçün DNS bazalarında TTL (time to live) adlı xüsusiyyət olur burada onun nə qədər müddətə keşdə qalmasına dair tövsiyə yazılır. Bu ədəd adətən 48 saata qədər olur. Ona görə domen adının İP ünvanlarını dəyişəndə tam funksional vəziyyətə gəlməsi üçün 48 saata qədər gözləmək tövsiyyə olunur.

```
; A Record
adil.az    3600    IN    A    54.229.111.35

; NS Record
adil.az    3600    IN    NS    mns01.domaincontrol.com.
adil.az    3600    IN    NS    mns02.domaincontrol.com.

; MX Record
adil.az    3600    IN    MX    10    aspmx.l.google.com.
adil.az    3600    IN    MX    20    alt1.aspmx.l.google.com.

; CNAME Record
blog    3600    IN    CNAME    domains.tumblr.com.
www     3600    IN    CNAME    adil.az.
```

Şəkil 2.6: DNS yazılarına aid nümunə. İkinci sütundakı 3600 – 3600 saniyə TTL-i bildirir.

Serverin İP ünvanı əldə edildikdən sonra ona müraciət edib saytı istəmək olar. İndi brauzer həmin serverə HTTP sorğusu göndərib saytın ilk səhifəsini əldə edir. Saytın HTML

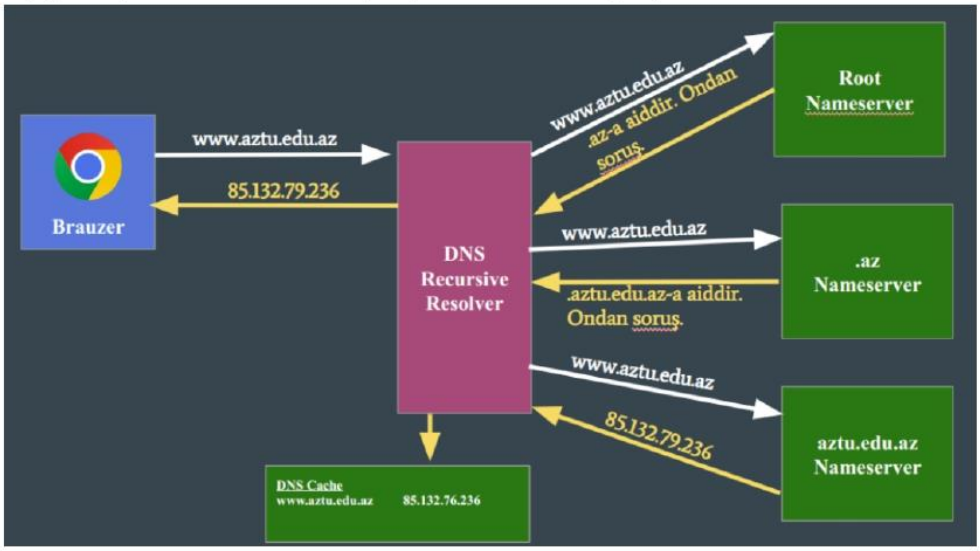
səhifəsində çoxlu sayda şəkillər, CSS fayllar, JavaScript elementləri və s. ola bilər. Brauzer onların hər biri üçün eyni qaydada HTTP sorğusu göndərir və özündə render edir.

Render prosesi üçün brauzer HTML kodu oxuyur, onu ağacvari data strukturuna çevirir. Daha sonra CSS kodlarını da istifadə edərək o struktura aid elementləri ekranda çəkir. Şəkillər varsa onu yükləyir və onu da səhifədə çəkir. Əgər JavaScript elementləri varsa JavaScript kodları səhifədə hər hansı bir elementi dəyişə bilir və render prosesi eyni ardıcıllıqla yenidən baş verir. Haqqında danışdığımız ağacvari struktura DOM – document object model deyilir. JavaScript vasitəsilə biz DOM üzərində əməliyyatlar yerinə yetirə bilirik.

DNS-in ələ keçirilməsi

Öyrəndik ki, DNS vasitəsilə ilə müştərilər (brauzer proqramı, tətbiqlər və s.) doman adlarının əsasında İP ünvanları əldə edir. Bunun üçün DNS resolver müəyyən qayda ilə lazım olan İP ünvanı tapır. Bu prosesi detallı şəkildə əvvəlki bölmədə izah etdik.

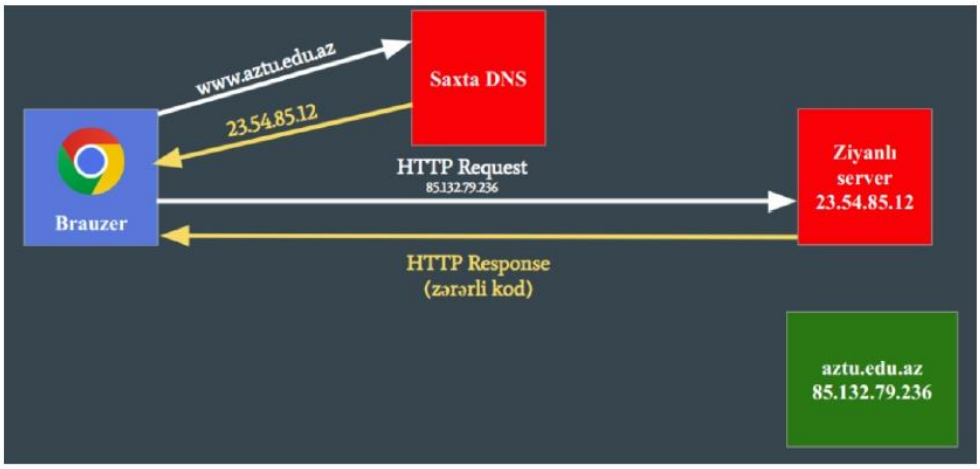
Aşağıdakı şəkildə sxematik olaraq bu proses öz əksini tapmışdır.



Şəkil 2.7: DNS resolverin iş prinsipi.

İndi təsəvvür edin ki, şəbəkədəki yönləndiricilərdən biri hakerlər tərəfindən ələ keçirilib. Və orada DNS resolver bilərəkdən dəyişdirilmişdir. Sxematik olaraq aşağıdakı şəkildə kimi bir mənzərə yaranacaqdır.

Belə bir hadisə baş verərsə zərərli serverlər müştərilərə ziyan vurmaq üçün bir çox imkanlara malik olacaqlar. Təbii ki, zərərverici proqramlarla sizin kompüterinizi və ya telefonunuzu yoluxdura bilərlər, fişing hücumu (növbəti dərslərdə ətraflı öyrənəcəyik) həyata keçirə bilərlər.



Şəkil 2.8: . DNS resolverin ələ keçirilməsi.

DNS-lərin ələ keçirilməsi müxtəlif yollarla baş verə bilər. Məsələn, hakerlər tərəfindən şəbəkə avadanlığının sındırılması. Ola bilsin ki, şəbəkə avadanlığındakı proqram təminatı (firmware) hansısa boşluğa malikdir. O boşluğu istifadə edərək hakerlər onu sındıra bilər. Və yaxud avadanlığın şifrəsi zavoddan çıxarkən verilmiş şifrədir (məsələn, admin/admin). Wi-fi quraşdırılmış bir çox ictimai yerlərdə təəssüf ki, bunun şahidi ola bilərsiniz.

Başqa bir səbəb isə şəbəkə avadanlığının sahibinin bu işi bilərəkdən etməsi ola bilər. Məsələn, ola bilər ki, hansısa ictimai obyektdə Wifi avadanlığının sahibi istifadəçilərə bilərəkdən zərər vurmaq istəyir. Ola bilər ki, hansısa bir tədbirdir və tədbir təşkilatçıları iştirakçılara zərər vurmaq istəyir. Ona görə də ictimai yerlərdə Wifi şəbəkəsinə qoşulmamaq yaxşı olardı.

Bu problemi həll etmək üçün üsullardan biri DNSSEC (Domain Name System Security Extensions) istifadə etməkdir. Amma bunun üçün provayderlər və operatorlar bu sistemi aktivləşdirməlidirlər. Hazırda heç də bütün provayderlər DNSSEC istifadə etmirlər.

3 | Mühazirə 3

Veb sayt nədir?

Biz hər gün internet üzərindən elektron poçtumuzu oxuyuruq, Whatsapp, Telegram kimi tətbiqlərlə kommunikasiya qururuq, elektron bank xidmətlərindən tətbiqlər vasitəsilə yararlanırıq. Bu cür saytlar, tətbiqlər bizim gündəlik həyatımızın parçasına çevriliblər. Bu mühazirə veb saytlar və onlara aid texnologiyalardan bəhs edir.

Veb saytlar

Veb sayt dedikdə adətən bir domen adı altında yerləşdirilən bir-biri ilə əlaqəli veb səhifələr toplusu nəzərdə tutulur. Məsələn, bsu.edu.az saytı Bakı Dövlət Universietinə məxsusdur və universitet haqqında müxtəlif məlumatları əks etdirən səhifələrdən ibarətdir. Bu səhifələr hiperlinklər vasitəsilə bir-birilə əlaqələndirilmişdir.

Şəkil 3.1: BDUnun saytının baş səhifəsi

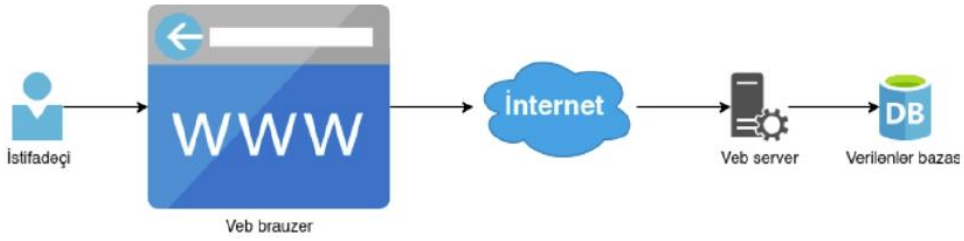
Veb saytları yaratmaq üçün müxtəlif texnologiyalar mövcuddur. Bunlara ümumi olaraq veb texnologiyalar deyirlər. Veb saytların yaradılması prosesinə veb proqramlaşdırma, o işlə məşğul olanlara veb proqramçı deyə müraciət edirlər. Müasir dövrdə texnologiyalar daha da geniş spektrə malik olduğundan bir veb saytı bir veb proqramçı adətən tək yaratmır. Onun fərqli hissələrinin üzərində fərqli şəxslər çalışır və beləlikdə backend proqramçı, front-end proqramçı, mobil tətbiq proqramçısı, fullstack proqramçı və s. kimi

yeni peşə adları hazırda geniş vüsət almışdır.

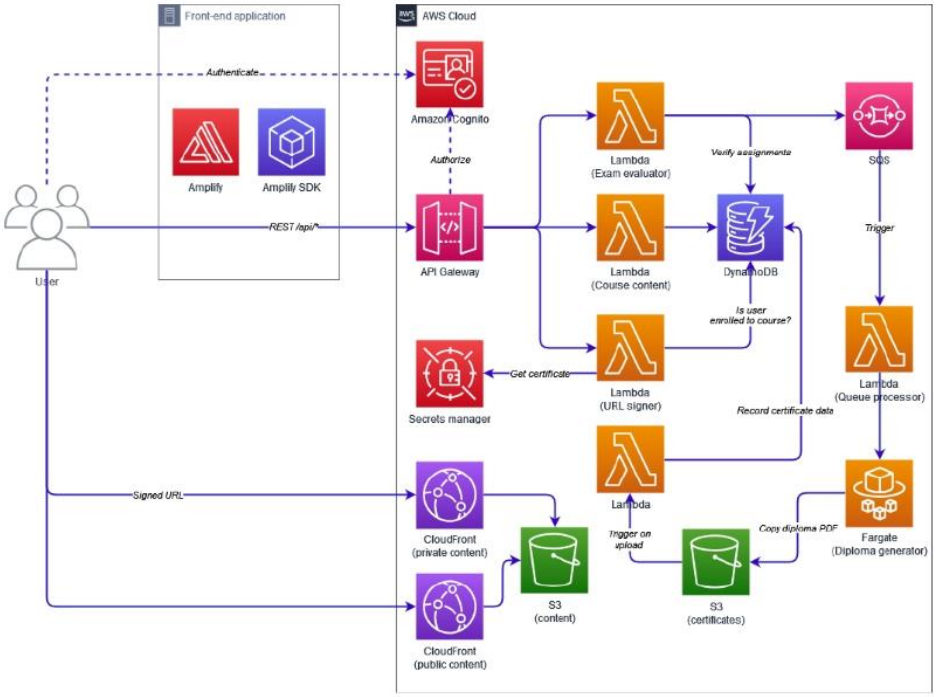
Veb texnologiyalar

Veb texnologiyaları veb-saytlar və veb tətbiqləri yaratmaq və idarə etmək üçün istifadə olunan alətləri və proqram təminatını əhatə edir. Bunlara aşağıdakılar daxildir:
HTML (HyperText Markup Language): Bütün vebsaytların onurğa sütunu olan HTML veb səhifələrin strukturunu və məzmununu təmin edir.
CSS (Cascading Style Sheets): CSS veb səhifələrin vizual görünüşü üçün istifadə olunur, müxtəlif cihazlara uyğunlaşan stilizasiya və dizaynları tərtib etməyə imkan verir. HTML ilə birlikdə istifadə edilir.
JavaScript: Veb səhifələrə interaktivlik əlavə edən, dinamik məzmun, animasiyalar və audio/video, oyunlar oynamaq və real vaxt məlumatlarını göstərmək kimi mürəkkəb funksiyaları təmin edən skript dilidir.
Veb serverləri: Veb məzmununu saxlayan, emal edən və istifadəçilərə çatdıran proqram və avadanlıqlar. Apache və Nginx məşhur veb serverlərə misaldır.
Verilənlər bazaları: Məlumatları saxlayan və idarə edən sistemlər. Veb saytlar istifadəçi məlumatlarını, yazıları, əməliyyatları və s. saxlamaq üçün verilənlər bazasından istifadə edir. MySQL, MongoDB və PostgreSQL geniş istifadə olunan verilənlər bazalarıdır.

Təbii ki, bu siyahını artırmaq olar. Amma baza anlayışları öyrənmək üçün yuxarıdakı siyahı fikrimcə yetərlidir. Veb saytlara daxil olarkən istifadəçi brauzeri açır və hər hansı bir ünvanı daxil edir. Brauzer proqramı HTTP protokolunu istifadə edərək internetdə həmin veb serverə müraciət edir. Veb server HTTP protokolunu başa düşən və HTTP sorğularına cavab verən proqram təminatıdır. Veb server müxtəlif proqramları da icra edə bilir. Bu proqramlar adətən verilənlər bazalarına qoşulur və məlumatları oradan əldə edir. Bu mexanizm aşağıdakı şəkildə sxematik olaraq verilmişdir.



Müasir dövrdə fundamental bloklar eyni qalsa da bir çox detallar daha fərqlidir. Məsələn, indi veb serverlər əvvəlki kimi bir fiziki serverdə deyil, virtual maşınlarda və ya konteynerlərdə ola bilir. Bu konteynerlər isə öz növbəsində bulud infrastrukturunda olur. Bulud xidmətləri eyni işi daha ucuz və səmərəli yerinə yetirmək üçün müxtəlif imkanlara malikdirlər. Məsələn, sadə bir onlayn öyrənmə veb saytının arxitekturası Amazon Web Services istifadəsi ilə aşağıdakı kimi ola bilər:



Veb tətbiqlər

Veb saytlar əvvəllər yalnız sadə veb səhifələr və onlar arasında əlaqələrdən ibarət idisə, sonralar daha interaktiv şəkil almağa başladı. Veb saytlar verilənlər bazaları ilə işləyə bilir, verilənləri o bazalarda yerləşdirir, oxuyur və emal edə bilirdi. Sonralar brauzerlər də öz növbəsində olduqca inkişaf etdilər. Brauzerlərin animasiyalar, videolar, 3D qrafika və s. kimi elementləri də vizuallaşdırmağa müvəffəq oldular. Həm server texnologiyalarının, həm də brauzer texnologiyalarının inkişafı veb tətbiqlərin meydana gəlməsinə səbəb oldu. Adi tətbiqlər təyin olunduğu əməliyyat sistemində çalışır. Məsələn, Microsoft Word proqram təminatı Windows əməliyyat sistemində quraşdırılırdı. Onun həm də Mac OS üçün başqa bir versiyası da var. Lakin indi onun veb versiyası brauzerin daxilində çalışır. Hansı əməliyyat sistemini istifadə etmədən onu yalnız brauzerin daxilində işə salmaq olar. Burada brauzerlər faktiki olaraq bir növ əməliyyat sistemi kimi çıxış edirlər. Bu ona gətirib çıxardı ki, insanlar biznes tətbiqlər əksəriyyətini artıq veb texnologiyaların köməyi ilə yazır və brauzerlərlə çalışdırırlar. Belə üsul ilə proqramları kompüterlərdə yükləməyə ehtiyac qalmır. Onları yeniləmək üçün kompüterlərə çıxışa da ehtiyac qalmır. Bir mərkəzdən yenilənmələri etmək olur. Kompüterlərə isə yalnız brauzer proqramı lazım olur. Bu cür proqram təminatlarına veb tətbiqlər deyirik. Bu dərsimizə qədər əsas anlayışları öyrəndik, veb saytlar, veb texnologiyalar, şəbəkə texnologiyalarına aid təsəvvürləri formalaşdırmaq üçün məlumatları qavramış olduq. Bundan sonrakı dərslər indiyədək

danışılan mövzuların detallarına varacaq və texniki məqamlarına həsr olunacaq.

URL

Əvvəlki dərslərdə dedik ki, insanın oxuya biləcəyi domen adı ilə saytlara müraciət edirik, sonra DNS onu İP ünvana tərcümə edir və biz İP ünvan vasitəsilə həmin serverə müraciət edib lazım olan məlumatı əldə edirik. Bəs server bizim hansı məlumatı istədiyimizi necə bilir? Məsələn, biz https://akta.az/az/about səhifəsinə daxil olduqda o AKTA haqqında məlumatı göstərir, https://akta.az/az/services səhifəsinə daxil olarkən isə AKTA təşkilatının göstərdiyi xidmətlərə aid məlumatları əks etdirir. Bizim brauzer DNS serverdən akta.az-a aid İP ünvanı istəyir. DNS server müvafiq İP ünvanı (İPv6 ilə 2606:4700:3034::ac43:d13e) bizə bildirir və bizim brauzer həmin İP ünvana müraciət edir. Hər iki müraciətdə sorğu eyni serverə gedir. Lakin server fərqli məlumatları göstərir. Bu necə baş verir?

Həmin İP ünvana malik server HTTP protokolunu başa düşür. Brauzerlər də HTTP protokoluna uyğun sorğular göndərirlər. Brazuer sorğunu həmin İP ünvana göndərəndə hansı səhifəni istədiyini də bildirir. Bu məlumat URL ünvanında əks edilmişdir. Server ona gələn sorğudakı URL ünvanına görə qərar verir ki, hansı məlumatı göstərsin.

URL veb-sayta daxil olmaq və ya onlayn fayla daxil olmaq üçün veb-brauzerinizə daxil etdiyiniz ünvandır. Bu, fiziki ünvanın rəqəmsal ekvivalentidir və sizi internetdə getmək istədiyiniz yerə istiqamətləndirir.

URL nədir? Onun quruluşu.

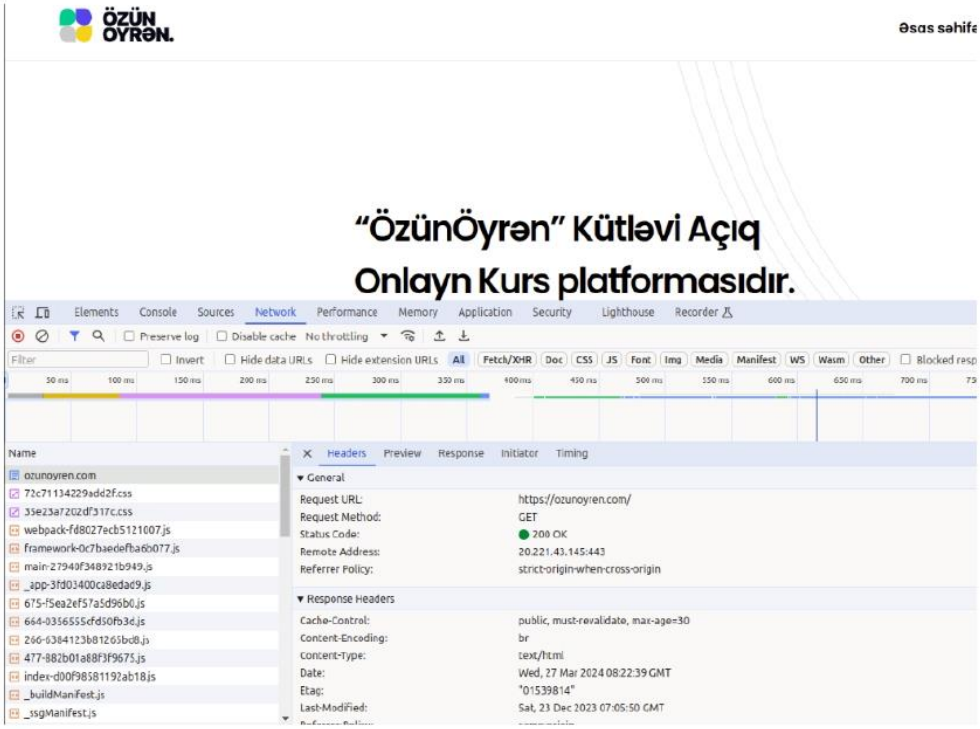
Şəkil 3.2: URL-in quruluşu

URL ünvanların strukturunu öyrənək. URL-in ilk hissəsində protokol bildirilir. Daha sonra hostname hissəsində onun serverinin ünvanı göstərilir. Bəzi hallarda iki nöqtə işarəsi qoyularaq port göstərilir. 80 və 443 uyğun olaraq HTTP və HTTPS üçün standart port nömrəsidir. Ona görə brauzerlərdə http ilə olanda birbaşa 80, https ilə olanda isə 443 qəbul edilir və brauzerlər onları ünvanlar panelində göstərmir. Daha sonra / (slash) işarəsi qoyulur və istiqamət (path) göstərilir. Bu sanki qovluqları bildirir. Bəzən sual işarəsi qoyulur və bu hissədə yazılan məlumata sorğu deyilir (query). Bəzi saytlarda siz # işa-

rəsi və ondan sonra bəzi məlumatları da görmüsünüz. Bu məlumat serverə göndərilmir. O yalnız brauzer tərəfindən istifadə edilir. Ona görə bəzi ədəbiyyatlarda bunu URL-in hissəsi sayırlar bəzi ədəbiyyatlarda isə saymırlar.

HTTP metodlar

HTTP metodları verilmiş resursda yerinə yetiriləcək fəaliyyəti müəyyən edir. Hər bir metodun xüsusi semantikası var və icra edilməli olan əməliyyat əsasında seçilir. Məsələn, yuxarıdakı nümunədə https://akta.az/az/about saytına brauzer vasitəsilə daxil olduqda o GET adlı metodu istifadə edir. GET metodu veb serverdən məlumatı əldə etmək üçün istifadə olunur. Bir veb saytı brauzer ilə açarkən o əvvəlcə səhifənin özünü yükləyir, daha sonra onun elementlərini yükləyir. Bununla GET metodunu istifadə edərək bəzən yüzlərlə faylı serverdən əldə edir. Bunu əyani görmək üçün brauzerinizin DevTools funksiyasını istifadə edə bilərsiniz.



Şəkil 3.3: Brauzerin DevTools funksiyası

Əgər siz hər hansı bir veb saytda bir formanı doldurub məlumat göndərdikdə o zaman POST, PUT və ya PATCH metodları istifadə edilir. Məsələn, sosial şəbəkədə bir şərh yazarkən brazuer o məlumatı lazım olan ünvana POST metodu ilə göndərir. Onun POST ilə göndərilməsi həmin veb tətbiqin və ya veb saytın kodunda müəyyən edilir. GET, POST,

PUT, PATCH, HEAD və s. bunlar HTTP protokolunda öz əksini tapmış metodlardır. Ən geniş yayılmış metodlar bunlardır:

GET

GET metodu müəyyən resursdan məlumat tələb etmək üçün istifadə olunur. Bu, əsasən serverdən veb səhifələri və ya digər məzmunu əldə etmək üçün istifadə edilən ən ümumi HTTP metodudur. GET sorğusu yalnız məlumatları əldə etməlidir və məlumatlara başqa heç bir təsiri olmamalıdır.

POST

POST metodu resurs yaratmaq və ya yadda saxlamaq üçün serverə məlumat göndərmək məqsədilə istifadə olunur. POST sorğuları tez-tez forma məlumatlarını göndərmək və ya faylları yükləyərkən istifadə olunur. POST sorğusu ilə göndərilən məlumatlar sorğunun mətninə daxil edilir. GETdən fərqli olaraq URL-in sorğu hissəsində deyil, ayrıca bölmədə göndərilir. Buna payload (yük) də deyilir. Server POST sorğusu görəndə onun payload hissəsini oxuyub emal edir.

PUT və PATCH

PUT və PATCH prinsip etibarı ilə POST kimi çalışır. Lakin bir çox hallarda semantik olaraq onları ayrırlar. Məsələn, PUT ilə mövcud məlumatı yeniləmək, PATCH ilə onun yalnız bir hissəsini yeniləmək, POST ilə məlumatı yaratmaq məqsədli istifadə edirlər. Lakin bu qayda onu istifadə edən proqramçının qərarından asılıdır.

DELETE

DELETE metodu göstərilən məlumatı silmək üçündür. Bu üsul sadədir - o, serverə URI tərəfindən müəyyən edilmiş resursu silməyi bildirir. Praktikada server həmişə sorğunu yerinə yetirməyə bilər (məsələn, resurs yoxdursa və ya istifadəçinin düzgün icazələri yoxdursa).

HEAD

HEAD metodu GET metoduna bənzəyir, lakin o, HEAD sorğusunun URI-si GET sorğusu olduğu halda qaytarılacaq başlıqları tələb edir. HEAD sorğuları GET sorğusu verməzdən əvvəl GET sorğusunun nə qaytaracağını yoxlamaq üçün faydalıdır, məsələn, resursun mövcud olub-olmadığını və əlçatan olub olmadığını yoxlamaq.

OPTIONS

OPTIONS metodu hədəf resurs üçün kommunikasiya imkanlarını öyrənir. Məsələn, POST sorğusu göndərilmədən OPTİONS metodu ilə sorğu göndərilir və serverə aid bəzi məlumatları əldə edir. Sonra brauzer və ya veb tətbiq onun cavabına əsasən POST sorğusunu formalaşdırır.

REST API

Yuxarıdakı metodlar faktiki olaraq məlumatı yaratmaq, oxumaq, dəyişdirmək, silmək funksiyalarını yerinə yetirir. Bunlara ingiliscə adlarının baş hərflərinə görə (create, read, update, delete) CRUD əməliyyatları dəsti də deyirlər. Onların hansı məlumatları yaratması, silməsi və ya dəyişdirməsi URLdəki elementlərə əsasən müəyyən edilir. Bir çox tətbiqlər bu qaydanı özləri üçün əsas götürüb. Bu qaydaları özündə birləşdirən məcmuya REST APİ deyirlər. REST standart və ya protokol deyil, şəbəkə proqramlarının dizaynı üçün üslubudur. Hazırda bir çox tətbiqlər REST üslublu veb-servislər üzərində qurulub.

Brauzer

Brauzerlər haqqında.

Brauzerlər haqqında bu dərsimizə qədər dəfələrlə qeyd etmişdik. İndi onlar haqqında daha geniş öyrənək. Brauzerlər istifadəçilər və internet arasında körpü rolunu oynayır, dünyanın hər yerindən veb səhifələri cihazınızın ekranına tərcümə edir və göstərir. Onlar istifadəçilərə bir-biri ilə əlaqəli veb təcrübəsini asanlaşdıraraq, hiperlinklər vasitəsilə bir veb-səhifədən digərinə keçməyə imkan verir.

Brauzer necə çalışır?

Bir URL daxil etməkdən veb səhifənin ekranına qədər olan proses aşağıdakı addımlardan ibarətdir:

1. Brauzerin ünvan panelində URL yazdığınız zaman brauzer hansı veb serverlə əlaqə saxlamağı başa düşmək üçün bu URL-i HTTP sorğusuna çevirir.
2. Brauzer domen adını internetdəki serveri müəyyən edən IP ünvanına çevirmək üçün DNS (Domain Name System) axtarışını həyata keçirir.
3. Brauzer IP ünvanı ilə əlaqəli veb serverə veb səhifənin məzmununu soruşan HTTP sorğusu göndərir.
4. Server sorğunu emal edir və CSS (Cascading Style Sheets), JavaScript, şəkillər və digər resurslarla birlikdə HTML (Hypertext Markup Language) formatında tələb olunan veb səhifə ilə cavab verir.

Əslində 4-cü addımdakı bütün fayllar birdən gəlmir. İlk olaraq yalnız sorğuda göstərilmiş HTML sənədi əldə edilir. O HTML sənədi CSS, JavaScript, şəkillər və s. haqqında öz daxilində məlumatları əks etdirir. Növbəti mühazirələrdə HTML, CSS və JavaScripti öyrənərkən bunları daha dəqiq araşdıracağıq. Brauzer HTML, CSS və JavaScript-i təhlil edir, DOM (Document Object Model) qurur və veb səhifəni ekranınızda göstərir. JavaScript interaktiv və dinamik veb səhifələrə imkan verməklə DOM-u dinamik şəkildə dəyişdirə bilər.

Veb servis

Veb servis nədir?

HTTP protokolu barədə qısa şəkildə əvvəlki mühazirədə bəhs etmişdik. Bu bölmədə bu protokol barədə daha ətraflı danışacaq və veb servislərdə HTTP protokolunun rolundan bəhs edəcəyik. İlkin olaraq gəlin veb servis nədir sualına cavab verək. Veb servis fərqli sistemlərin və ya tətbiqlərin bir-birləri ilə əlaqəsini və məlumat mübadiləsi aparmağını təmin edən vəsitədir. Veb servislərdən istifadə edərək tətbiqlər fərqli platformalarda və fərqli qurğularda olmasına baxmayaraq ,şəbəkə vasitəsilə rahatlıqla əlaqə saxlaya və məlumat mübadiləsi apara bilərlər.

Misal üçün, götürək hava haqqında məlumat servislərini. Hər birimizin telefonunda hava haqqında məlumatı göstərən bir tətbiq var və biz istədiyimiz vaxt istədiyimiz bir yer üçün bu gün və ya növbəti günlərə olan hava vəziyyətinə baxa bilirik. Bəs bu necə mümkündür? Çox sadə. Siz telefonunuzda həmin tətbiqi açdığınız vaxt, telefonunuz veb servis vasitəsilə bu məlumatları əldə edir və sizə göstərir. Bunun üçün hava tətbiqi əvvəlcə yer və vaxt haqda məlumatları həmin servisə ötürür, həmin servis isə öz növbəsində uyğun məlumatları geri göndərir və siz həmin məlumatları telefonunuz ekranında görürsünüz. Daha bir nümunə olaraq, biz ödəniş sistemlərini göstərə bilərik. Deyək ki, siz onlayn kitab mağazasında bir neçə kitabı səbətinizə atmısınız və ödəniş etmək istəyirsiniz. Ödənişlər əksər hallarda həmin onlayn mağazanın özü vasitəsilə həyata keçirilmir, yəni ödənişi həyata keçirən qurum tamamilə başqa olur. Deyək ki, siz A saytında bazarlıq edirsiniz, ödənişlərdən isə B saytı məsuliyyət daşıyır. Burada B saytı hər hansı bir bank ola bilər. Siz kart məlumatlarınızı daxil etdikdən sonra B saytı sizin kartınızdan pulu çıxır və veb servislər vasitəsilə A saytına ödənişin uğurlu və ya uğursuz olması ilə bağlı məlumatları ötürür. Beləliklə siz istədiyiniz kitabları əldə etmiş olursunuz.

Veb servislər ilə məlumat mübadiləsi etmək üçün mütləq şəkildə bir şəbəkə lazımdır. Siz əksər hallarda hər hansı bir mobil və ya lokal şəbəkəyə qoşulu olursunuz və məlumat mübadiləsi bu şəbəkə üzərindən həyata keçirilir. İndi sual yaranır ki, bəs iki qurğu - deyək ki, bir telefon ilə bir planşet - bir-biri ilə şəbəkə üzərindən necə əlaqə saxlayır və bir-birini tam olaraq necə başa düşür? Bunun üçün protokollar mövcuddur. Veb servislər əksər hallarda HTTP protokolu ilə çalışır. Bu dərsimizdə HTTP ilə REST tipli veb servislər haqqında danışacağıq.

HTTP protokolu necə çalışır?

Əvvəlki mühazirədə HTTP protokolu barədə ümumi məlumat verilmişdi. İndi gəlin onu bir qədər daha detallı öyrənək. HTTP necə işləyir? Adi kağız məktubu poçt ilə necə göndərirsinizsə HTTP də ona oxşayır. Təsəvvür edin ki, keçmiş zamandasınız və poçt xidməti vasitəsilə dostunuza məktub göndərirsiniz. Zərfin üzərinə bir ünvan yazırsınız, onu poçt şöbəsinin qarşısındakı poçt qutusuna atırsınız.

Müəyyən vaxt keçdikdən sonra poçt şöbəsinin işçisi sizin məktubu alır və onu başqa bir poçt şöbəsinə aparır.

Başqa poçt şöbəsinin işçisi isə onu növbəti ünvana təhvil verir.

Bu minvalla sizin məktub müxtəlif poçt şöbələrini gəzir.

Nəhayət gəlib son ünvanına sizin dostunuzun ünvanına çatdırılır.

Dostunuz məktubu oxuyur və sizə bir cavab yazır. Sonra onu poçt şöbəsinə təhvil verir və eyni qayda ilə poçt şöbələrini gəzərək məktub sizə gəlib çatır. Məktubun bir poçt şöbəsindən digərinə getməsi, sonra cavabın oxşar qayda ilə geri qayıtması ideyası elə HTTP protokolu ilə məlumatın bir mənbədən digərinə gedib çatmasına bənzəyir.

HTTP internetdə sənədlərin mübadiləsini tənzimləyən protokol – qaydalar toplusudur. Brauzerinizə ünvan yazdığınız zaman bu, məktubunuza təyinat ünvanını yazmağa bənzəyir. Brauzeriniz məktubu poçt qutusuna atmaq kimi bir HTTP sorğusu göndərir. Bu sorğu internet üzərindən vebsaytın serverinə gedir. Məsələn, Siz ÖzünÖyrən platformasından video dərsi dinləmək üçün ozunoyren.com ünvanını brauzerinizə daxil etdikdən sonra "Enter" düyməsini vurduğunuzda nə baş verdiyini addım-addım öyrənək:

DNS: Əvvəlcə sizin brauzer DNS protokolunu istifadə edərək ÖzünÖyrənin serverinin İP ünvanını müəyyənləşdirir. Bağlantının qurulması: Brauzeriniz TCP/IP (Transmission Control Protocol/Internet Protocol) istifadə edərək serverə qoşulmağa başlayır. HTTP Sorğunun Göndərilməsi: Brauzer serverə veb səhifəni tələb edən HTTP sorğusu göndərir.

Server cavabı: Server bu sorğunu alır, emal edir və cavabı geri göndərir. Bu, HTTP formatında səliqəli şəkildə paketlənmiş, tələb etdiyiniz veb səhifədir.

Tələb olunan səhifə əlçatandırsa, "200 OK" status mesajı alırsınız. Əks təqdirdə, '404 Tapılmadı' əldə edə bilərsiniz. Burada 200 və 404 xüsusi status kodlarıdır. HTTP protokolunun bir sıra status kodları var. Düşünürəm ki, hamısını sadalamağa və əzbərləməyə ehtiyac yoxdur. Bircə onu yadda saxlayın ki, 1 ilə başlayan kodlar məlumatverici kodlar-

dır, 200 və 2 ilə başlayan digər status kodlar əməliyyatın uğurlu baş tutduğunu bildirən kodlardır, 3 ilə başlayan kodlar yönləndirmə olduğunu bildirir, 4 ilə başlayanlar müştəri xətasını bildirir, 5 ilə başlayanlar isə serverdə hansısa xəta olduğunu bildirən kodlardır. Brauzer Veb Səhifəni göstərir: Brauzeriniz HTTP cavabını alır, və veb səhifəni və səhifədəki bu videonu göstərir.

Hər dəfə veb servis ilə qarşılıqlı əlaqədə olanda – istər tvitterdə status paylaşanda, istər xəbərləri oxuyanda, istər bu videonu dinləyəndə – siz veb serverlərlə əlaqə saxlamaq üçün HTTP-dən istifadə edir və yüzlərlə sorğu göndərirsiniz.

JSON

Əvvəlki dərsdə siz HTTP protokolu haqda öyrəndiniz. HTTP protokolunu istifadə edərək qurğular bir-birləri ilə məlumat mübadiləsi aparırlar. Başqa bir məsələ mübadilə etdiyimiz məlumatın formatıdır. HTTP protokolu ilkin olaraq hipermətnləri mübadilə etmək üçün yaradılmışdır. Hipermətnə nümunə olaraq, HTML mətnini göstərmək olar. HTML mətnləri qaydaya salmaq üçün istifadə olunan bir markap dilidir və internetdə gördüyünüz bütün vebsaytların strukturu HTML ilə qurulmuşdur.

Siz hər hansı bir vebsayta daxil olduğunuzda vebserverə HTTP GET sorğusu göndərilir. GET sorğusu hər hansı bir məlumatı əldə etmək üçün nəzərdə tutulmuş bir HTTP felidir. Server isə bunun cavabı olaraq HTTP 200 OK mətni göndərir, yəni hər şeyin qaydasında olduğunu bildirir. Bununla yanaşı cavab mətni də göndərilir. Bu halda cavab mətni daxil olduğunuz vebsaytın HTML ilə yazılmış strukturudur. Server bundan əlavə cavab mətninin nə formatda olduğunu da bildirməlidir, çünki brauzeriniz alınan mətnin HTML yoxsa başqa bir formatda olduğunu başa düşməlidir. Bunun üçün server Content-Type başlığı altında text/html sətrini də brauzerə yollayır.

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

Texnologiyanın inkişafı və yeni ehtiyacların yaranması ilə insanlar HTTP-ni tək hiper-

mətlərin mübadiləsi üçün yox başqa formatda məlumatların mübadiləsi üçün də istifadə etməyi qərara aldılar. Yəni əgər server HTML formatında yox, başqa formatda mətn göndərsə, onu Content-Type başlığı altında sorğunu göndərən tərəfə bildirməlidir. Bu dərsdə biz çox geniş yayılmış bir mətn formatı olan JSON-u başa düşməyə çalışacağıq. JSON, JavaScript Object Notation sözlərinin qısaldılmış formasıdır və ilkin olaraq JavaScript proqramlaşdırma dilindəki obyektlərdən ilhamlanaraq yaradılmışdır. JSON-nun əsas özəlliklərindən biri isə odur ki, o insanlar üçün aydın başa düşülən bir formatdır. Deyək ki, siz telefonunuzda LinkedIn tətbiqinə daxil olub öz profil səhifənizə keçirsiniz. Bu zaman LinkedIn tətbiqi profil məlumatlarından məsul olan veb servisə HTTP GET sorğusu göndərib sizin profiliniz haqda məlumat alacaq. Veb servisin cavabı isə JSON formatında təxminən belə bir şey olacaq:

```
{
  "first_name": "John",
  "last_name": "Doe",
  "title": "Software Engineer",
  "current_company": "Google",
  "education": "Harvard University",
  "skills": [
    "Python",
    "Data Analysis",
    "Data Structures"
  ]
}
```

JSON formatı da lüğətlər kimidir; o açar sözlərindən və dəyərlərdən ibarətdir. Burada göstərdiyimiz nümunə təbii ki, sadələşdirilmiş və xəyali bir nümunədir. LinkedIn tətbiqində profilinizə daxil olduğunuzda yəqin ki, bundan dəfələrlə çox məlumat sizin telefonunuza cavab olaraq göndərilir. Buraya şəkillər, hazırki iş yeriniz, təcrübələriniz, təhsiliniz, sertifikatlarınız və bacarıqlarınız da daxildir. Veb servis cavab mətni olaraq JSON göndərdiyinzə sorğunu göndərən tərəfə bunu Content-Type başlığı altında bildirməlidir. Bunun üçün Content-Type: application/json başlığı göndərilir və sorğunu göndərən tərəf mətnə JSON kimi yanaşmalı olduğunu başa düşür. Olduqca sadədir, elə deyilmi? Əslində mürəkkəb və qarışıq kimi tanıdığımız sistemlər bu cür sadə protokollarla bir-birini başa düşür və məlumat paylaşırlar.

Digər mübadilə formatları (YAML, XML)

Məlumat mübadiləsi aparmaq üçün fərqli formatlar mövcuddur. Əvvəlki dərsdə biz JSON haqda danışqıq və qısaca olaraq HTML formatına da toxunduq. JSON hazırda ən çox istifadə olunan və geniş yayılmış formatlardan olmasına baxmayaraq bəzi veb servislər XML və YAML kimi digər formatları da dəstəkləyə bilər. Gəlin XML-dən başlayaq. XML - ekstensiv markap dili mənasına gəlir və JSON-dan əvvəl çox məşhur bir format idi. Əvvəlki videoda gördüyümüz JSON nümunəsini XML formatında göstərək və onu incələyək:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <first_name>John</first_name>
    <last_name>Doe</last_name>
    <title>Software Engineer</title>
    <current_company>Google</current_company>
    <education>Harvard University</education>
    <skills>Python</skills>
    <skills>Data Analysis</skills>
    <skills>Data Structures</skills>
</root>
```

Gördüyünüz kimi XML formatını ilkin baxışdan başa düşmək bir az çətindir. XML-də əsas anlayışlardan biri teqlərdir. JSON-da açar söz kimi yazılan "first_name" burada $< first\_name >$ teqi olaraq yazılır və həmin açar sözün dəyəri açılış və bağlanış teqinin arasında qeyd olunur. Eyni qaydada, last_name, title və digər atributlar da onun kimi. Burada sol tərəfdə gördüyümüz teqlər açılış teqi, sağda gördüyümüz isə bağlanış teqidir.

Əgər iki tətbiq bir-biri ilə XML formatında mübadilə aparısa onlar Content-Type başlığı altında application/xml göndərirlər ki, məlumatın XML formatında olduğunu başa düşsünlər. Müasir dövrdə XML çox geniş yayılmış bir format olmasa da bəzi köhnə veb servislərdə onunla rastlaşa bilərsiniz.

Danışmalı olduğumuz digər format isə YAML-dır. YAML - yet another markup language - yəni başqa bir markap dili anlamına gəlir. YAML formatı XML ilə müqayisədə daha rahat başa düşüləndir:

```yaml
first_name: John
last_name: Doe
title: Software Engineer
current_company: Google
education: Harvard University
skills:
  - Python
  - Data Analysis
  - Data Structures
```

YAML da öz növbəsində JSON-da olduğu kimi açar sözlərdən və dəyərlərdən isitfadə edir. JSON ilə arasındakı fərq ondan ibarətdir ki, burada nə açar sözlərini, nə də ki dəyərləri dırnaq işarəsi içərisində yazmaq məcburiyyətində deyilik. Bundan əlavə YAML formatı fiqurlu mötərizə əvəzinə boşluq simvollarından istifadə edir. Nümunə olaraq, skills açar sözünün altındakı dəyərlər skills sözünün olduğu sıradan iki boşluq sağ tərəfə sürüşdürülmüşdür. Bununla bəlli olur ki, Python, Data Analaysis və Data Structures dəyərləri skills açar sözünün altındadır.

REST API anlayışı

REST API günümüzdə çox istifadə olunan bir termindir. Əvvəlcə gəlin API sözünü başa düşməyə çalışaq. API sözü - Application Programming Interface - yəni tətbiqi proqramlaşdırma interfeysi mənasına gəlir. API ilə istifadəçi interfeysi eyni şey deyil, bunu qarışdırmayaq. API daha çox iki kompüter tətbiqinin bir-biri ilə əlaqə qurma vasitəsidir. İllər öncə bu həm də kompüterin proqramları arasında əlaqə qurmaq üçün istifadə olunsa da son zamanlar, Veb servislərlə sinonim olmağa başlayıb. API-ları daha yaxşı başa düşmək üçün gəlin real həyatdan bir nümunəyə baxaq.

Deyək ki, siz hər hansı bir restorana getmisiniz və yemək sifariş etmək istəyirsiniz. Restoranın mətbəxində kartof, soğan, sarımsaq və ət kimi müxtəlif ərzaq növləri mövcuddur. Siz bir başa mətbəxə gedib aşpaza istəklərinizi çatdıra bilərsiniz. Məsələn, gedib deyirsiniz ki, yağı tök tavaya, soğanı doğra, əti qızart və s. Amma heç bir restoranda belə bir şey etmirik elə deyilmi? Əgər elə olsa idi, həqiqətən də çox qəribə olardı.

Əvəzində siz masanızda əyləşdiyiniz zaman ofisiant sizə yaxınlaşır və menyunu sizə göstərir. Menyuda isə əsas yeməklərin, məzələrin və şorbaların adları var. Siz siyahıdan istədiyiniz yeməkləri seçib ofisianta bildirirsiniz, ofisiant isə öz növbəsində bunu mətbəxə bildirir. Mətbəxdəki aşpazlar isə sifarişinizi hazırlamağa başlayırlar. API sözündəki application yəni tətbiq burada aşpazlardır. Biz onları proqramlaşdırmaq üçün bir interfeysdən istifadə etdik, yəni ofisiantdan. İnterfeys bizim qarşılaşdığımız və komandaları verdiyimiz komponentdir. Qısası, biz mətbəxə gedib aşpaz ilə danışmaq əvəzinə ofisiantın bizə təqdim etdiyi qəşəng və səliqəli menyuya baxıb öz seçimimizi etdik və bizim istəklərimiz gedib aşpaza çatdı.

Eyni qaydada, fərqli proqram təminatı sistemləri bir-birləri ilə əlaqə saxlamaq üçün API-lardan istifadə edir. Əgər bir tətbiq digər tətbiqin hər hansı bir xüsusiyyətini istifadə etmək istəyirsə, digər proqramın verdiyi interfeysdən yəni onu API-ndan istifadə edir. Misal olaraq, siz Pythonda öz proqram təminatınızı yaza bilərsiniz və sizin proqramınız fərqli sistemin, misalçün Google Mapsın verdiyi API-lardan istifadə edə bilər. Və ya siz Python ilə 3D oyun yazırsınızsa, bu zaman 3D qrafikləri göstərə bilmək üçün kompüterinizin qrafik prosessorunun sizə verdiyi API-lardan istifadə edirsiniz.

Yaxşı, API sözünü başa düşdük, bəs REST API nə deməkdir? REST API-lar tamamilə HTTP protokolu üzərindən çalışan API-lardır. Yəni siz internetdəki hər hansı bir tətbiqin funksionallıqlarını istifadə etmək istəyirsinizsə həmin sistemin sizə verdiyi REST API-dan yararlana bilərsiniz. Misalçün, siz öz tətbiqinizdə Twitterdə paylaşılan tvitlər arasında axtarış etmək istəyirsinizə https://api.twitter.com/2/tweets/search/all URL-inə HTTP GET request göndərə bilərsiniz. Gördüyünüz kimi REST API-lar sadəcə HTTP protokolunu istifadə edən və müəyyən standartlara uyğun olan API-lardır. Onu da qeyd edək ki, REST API-lar əksər hallarda JSON formatında məlumat mübadiləsi aparırlar. Belə ki, biz müştərilər haqda məlumatı JSON olaraq alırıq və yeni müştəri yaratmaq istədiyimizdə onun məlumatlarını da JSON formatında serverə göndəririk.

4 | Mühazirə 4

HTML

Veb təhlküsizliklə məşğul olan mühəndislər saytların yaradılması ilə məşğul olmurlar. Onlar saytları müdafiə etmək və müdafiəni öyrənmək üçün sındırmaqla məşğul olurlar. Lakin veb sistemləri qoruya bilmək üçün, onların təhlükəsizliyini təmin edə bilmək üçün istifadə olunan texnologiyaları mükəmməl şəkildə öyrənmək vacibdir. Ona görə də bu mühazirədə və növbəti mühazirədə HTML, CSS və JavaScriptdən bəhs edəcəyik.

HTML veb səhifələrin əsasını təşkil edir və onu bilmək veb səhifələrə zərərli skriptlərin əlavə olunduğu Cross-Site Scripting (XSS) hücumları kimi təhlükəsizlik problemlərini aşkar etməyə və həll etməyə kömək edir. CSS sadəcə saytların yaxşı görünməsini təmin etmək kimi görünə bilər, lakin o, həm də təhlükəsizliklə bağlıdır. CSS biliyi, istifadəçilərin fərqinə varmadan zərərli bir şeyə kliklləmək üçün aldadıldığı klikləmə kimi hücumların qarşısını almaq üçün lazımdır. CSS vasitəsilə zərərli HTML elementləri istifadəçidən gizlədilə bilər. JavaScript isə veb-saytları interaktiv edir, lakin təhlükəsizlik riskləri də təqdim edə bilər.

JavaScript ilə tanış olmaq təhlükəsizlik mühəndislərinə skriptlərdən yaranan zəifliklər müəyyən etməyə və aradan qaldırmağa imkan verir. Bir sözlə, veb təhlükəsizlik mühəndisləri veb-saytları müxtəlif kibertəhlükələrdən effektiv şəkildə qorumaq üçün bu texnologiyaları başa düşməlidirlər. Bu bilik onlara veb səhifələrin strukturuna, görünüşünə və interaktiv xüsusiyyətlərinə xas olan zəifliklər aradan qaldırmaqla daha təhlükəsiz veb mühitləri qurmağa imkan verir.

İlk olaraq HTMLdən başlayacağıq.

HTML nədir?

HTML (Hypertext Markup Language) – veb saytların yaradılması üçün istifadə olunan hipermətn işarələmə dilidir. HTML proqramlaşdırma dili deyildir, çünki html kodları ilə öz başına işləyən bir proqram yazmaq olmaz. Brauzer ilə hər hansı bir səhifəyə daxil olduqda, ekranda gördüyümüz bütün hər şey HTML dilində yazılmışdır. Bu hipermətn dili – veb sehifələrdə mətn bloklarının, təsvirlərin, multimediya elementlərinin (video, musiqi və s.), cədvəllərin qurulmasına, hiper-istinadların, və bu zəmindən olan elementlər istifadə etməyə imkan yaradır.

HTML-də elementlər.

HTML dilində əsas element teqlərdir (ing.tag). HTML dilində teqlər “<” və “>” işarələri arasında yazılır. Məsələn,

```
<h1>Salam</h1>
```

Bu işarələmə “Salam” mətnini böyük ölçüdə yazacaqdır. Burada “h1” teqi birinci dərəcəli başlıq mənasını verir. Teqlər <h1> şəklində açılır, və </h1> şəklində bağlanır. Teqlərin arasında isə lazım olan mətn elementi verilir, yuxarıdakı nümunədə Salam mətnidir. İndi HTML sənədinin strukturu ilə bağlı daha ətraflı öyrənək. Öyrənmə prosesi praktik olsun deyə yazılanları yerinə yetirmək yaxşı olardı. VS Code mətn redaktorunu açıb aşağıdakı mətni yazın:

```
<h1>Salam</h1>
```

Mətni yadda saxlayın və faylın genişlənməsini “.html” kimi qeyd edin və o faylı brauzer (Google Chrome, Firefox, Edge, Brave və s.) ilə açın. Hər dəfə yeni dəyişiklik etdikdə faylı yadda saxlayın və brauzeri yeniləyin (“Refresh” edin). Html sənədi <!DOCTYPE html> kodu ilə başlayır. Bu işarələmə sənədin HTML standartına uyğun olacağını bildirir. Bundan sonra sənəd <html> teqi ilə açılır və </html> teqi ilə bağlanır.

Onun daxilində <head>...</head> teqi arasında səhifənin başlığı və metadata (bu barədə növbəti mövzularda ətraflı danışacağıq) təsvir edilir. HTML sənədinin görünən hissəsi, yəni gövdəsi <body>..... </body> teqi arasında yazılır.

```
<!DOCTYPE html>
<html>
<head>
    <title>Bu səhifənin başlığıdır.</title>
</head>
<body>
    Bu səhifənin gövdəsidir.
</body>
</html>
```

HTML elementlərin atributları

HTML atributları elementlər haqqında əlavə xüsusiyyətləri təyin edir. Atributlar həmişə açılış teqində göstərilir. Atributlar adətən ad/dəyər kimi təqdim olur: name=”value”. Məsələn, <a> teqi hiperlink (keçid) üçün istifadə edilir və onun href atributu keçidin getdiyi

səhifənin URL-ni təyin edir:

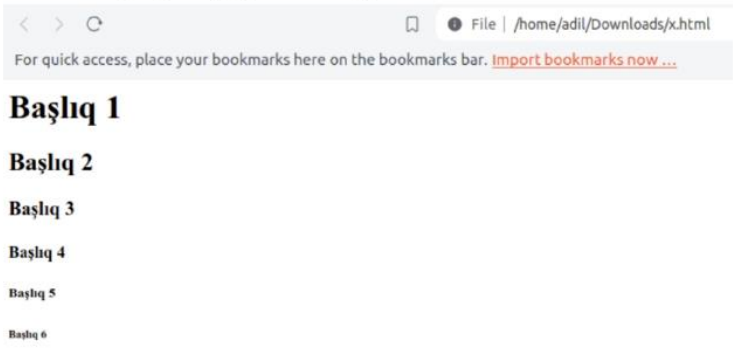
```
<a href="https://www.ozunoyren.com">ÖzünÖyrən</a>
```

Geniş istifadə edilən teqlər.

İndi HTMLdə geniş istifadə edilən teqlərin bir hissəsini öyrənək. Başlıq elementləri Başlıq elementləri html dilində <h1>text </h1>, ...<h6>text</h6> kimi yazılır edilir. H1 -H6 arası olan başlıq elemenlərindən ən böyüyü <h1>text</h1> , ən kiçiyi isə <h6>text</h6> -teqidir.

```
<h1>Başlıq 1</h1>
<h2>Başlıq 2</h2>
<h3>Başlıq 3</h3>
<h4>Başlıq 4</h4>
<h5>Başlıq 5</h5>
<h6>Başlıq 6</h6>
```

Bu faylı brauzer ilə açdıqda aşağıdakı kimi görünəcək:



Paraqraf elementi

<p> teqi -html dilində paraqrafları yazmaq üçün istifadə edilir. Bir qayda olaraq paraqraflar arasında müəyyən məsafə mövcud olur. Lakin, CSS vasitəsilə o məsafələri dəyişmək mümkündür. Paraqrafın nəticəsi həmişə yeni sətrdə başlayır və brauzer avtomatik olaraq mətinin əvvəlinə və sonuna boşluq işarəsi əlavə edir.

```
<p>Bu paraqrafdır.</p>
<p> Bu isə başqa bir paraqrafdır.</p>
```

"Span" elementi

HTML -də   teqi bir növ mətnlər üçün konteyner xaraketerini daşıyır. Adətən mətnlər üçün istifadə olunsa da daxilində başqa elementlər də (şəkillər, keçidlər və s.) ola bilər.

Siyahılar

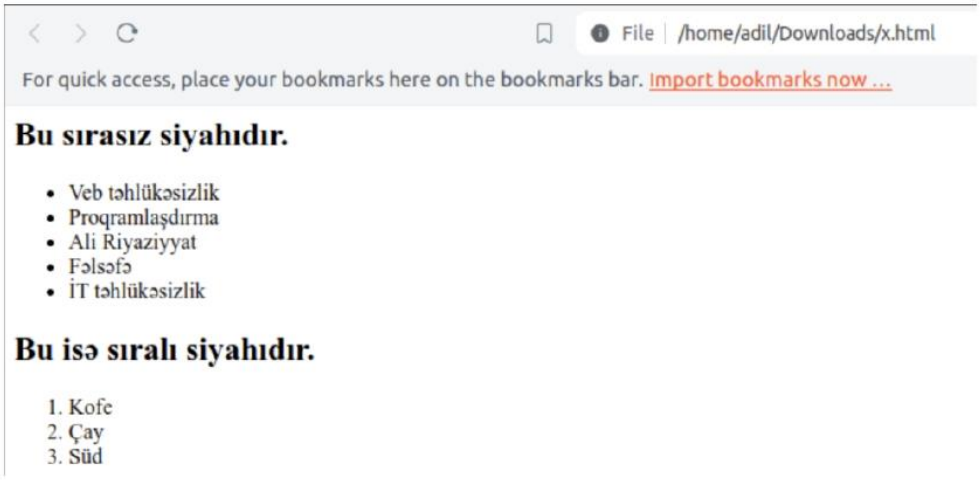
HTML də siyahılar iki cür olur: Sıralı və ya sırasız. (ordered vs unordered). Sırasız siyahı üçün <ul> teqindən istifadə olunur. Ul ingilis dilində "unoredered list" sözlərinin baş hərflərini bildirir. Uyğun olaraq OL teqi "orderet list" sözlərinin baş hərflərindən təşkil edilib. Siyahı yaratmaq üçün <ul> teqinin daxilində teqindən istifadə olunur.

```
<h2>Bu sırasız siyahıdır.</h2>

<ul>
  <li>Veb təhlükəsizlik</li>
  <li>Proqramlaşdırma</li>
  <li>Ali Riyaziyyat</li>
  <li>Fəlsəfə</li>
  <li>İT təhlükəsizlik</li>
</ul>


<h2>Bu isə sıralı siyahıdır.</h2>
<ol>
  <li>Kofe</li>
  <li>Çay</li>
  <li>Süd</li>
</ol>
```

Bu kodun nəticəsi brauzerdə belə görünəcək:



 siyahıları üçün siyahını elementlərinin qarşısındakı işarəni dəyişmək mümkündür. Bunun üçün onun "list-style" xassəsindən istifadə edilir. Bu xassəni CSS vasitəsilə style attributunun daxilində yazmaq lazımdır. CSS haqqında bir qədər sonra öyrənəcəyik.

```
<ul style="list-style: square;">
  <li>Veb təhlükəsizlik</li>
  <li>Proqramlaşdırma</li>
  <li>Ali Riyaziyyat</li>
  <li>Fəlsəfə</li>
  <li>İT təhlükəsizlik</li>
</ul>
```

Bu xassəni istifadə edərək siyahının qarşısındakı işarələri hətta yığışdırmaq, istədiyiniz mətni əlavə etmək, istədiyiniz şəkil istifadə etmək də olar. Daha ətraflı öyrənmək üçün https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type saytındakı sənədlərə nəzər yetirə bilərsiniz.

Div, header və footer elementləri

<Div> elementi verilən məzmun üçün təyin olunmuş ümumi konteynerdi. Mahiyyətcə heç nəyi təmsil etmir, sadəcə bir blok yaradır və yaxud bir neçə təqi bir blokun daxilinə salır. Bu teq ən çox istifadə olunan teqlərdən biridir, çünki web saytlarda elementləri bir-birindən ayırmağa kömək edir. İstənilən veb sayta daxil olduqda siz onun kodalrına baxsanız çoxlu sayda div teqlərindən ibarət olduğunu görəcəksiniz.

```
<div>
    <h2>Bu div elementində bir başlıqdır.</h2>
    <p>Bu div elementində bir paraqrafdır.</p>
</div>
```

Div teqi blok səviyyəli teqdir. Hər yeni div eyni sətirdə deyil, yeni sətirdən başlayır. Lakin bu xassəni CSS vasitəsilə dəyişmək mümkündür. <header>, <footer>, <nav>, <article> HTML-in 5-ci versiyasında peyda olmuş elementlərdir. Məğz etibarı ilə div-dən fərqlənmir. Sadəcə olaraq səhifələrin semantikası və kodların oxunaqlığı üçün istifadə edilir. Məsələn, axtarış sistemləri bloq səhifələrində <article> elementinin daxilində məqalənin mətninin yazıldığını "düşünür".

img elementi

HTML dilində <img> teqi şəkil əlavə etmək üçün istifadə olunur. img teginin iki əsas atributu var. src - Şəkilə gedən keçidi, yolu müəyyən edir, alt- müxtəlif səbəblərdən yaranan problem zamanı, şəkil göstərilməsində çətinlik baş verərsə, şəkil üçün mətn göstərirdi. Bir növ şəkilin nəyə aid olduğunu göstərmək üçün verilən izahdır.

```
<img
  src="şəkillər/aztu.jpg"
  alt="Azərbaycan Texniki Universiteti logo"/>
```

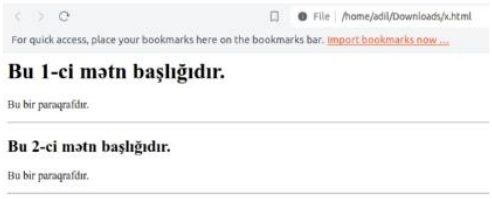
İMG teqinin ölçülərini CSS vasitəsilə dəyişmək olar. Ölçülər verilmədikdə o şəkilin orijinal ölçüsünü qəbul edir. Bəzi saytlarda ölçüləri 0 piksel təyin edib şəkili görünməz hala salırlar. Belə olduqda şəkil görünməsə də o yenə də brauzer tərəfindən yüklənir. Adətən istifadəçilərin izlənməsi və ya cookielərin (sonrakı mühazirələrdə öyrənəcəyik) oğurlanması üçün istifadə olunan üsuldur.

Br və Hr elementləri

<hr>- teqi html sənədində kontentin ayrılması üçün istifadə olunur.

```
<h1>Bu  1-ci mətn başlığıdır.</h1>
<p>Bu bir paraqrafdır.</p>
<hr />
<h2>Bu 2-ci mətn başlığıdır.</h2>
<p>Bu bir paraqrafdır.</p>
<hr />
```

Bu kodun nəticəsi aşağıdakı kimi olacaq:

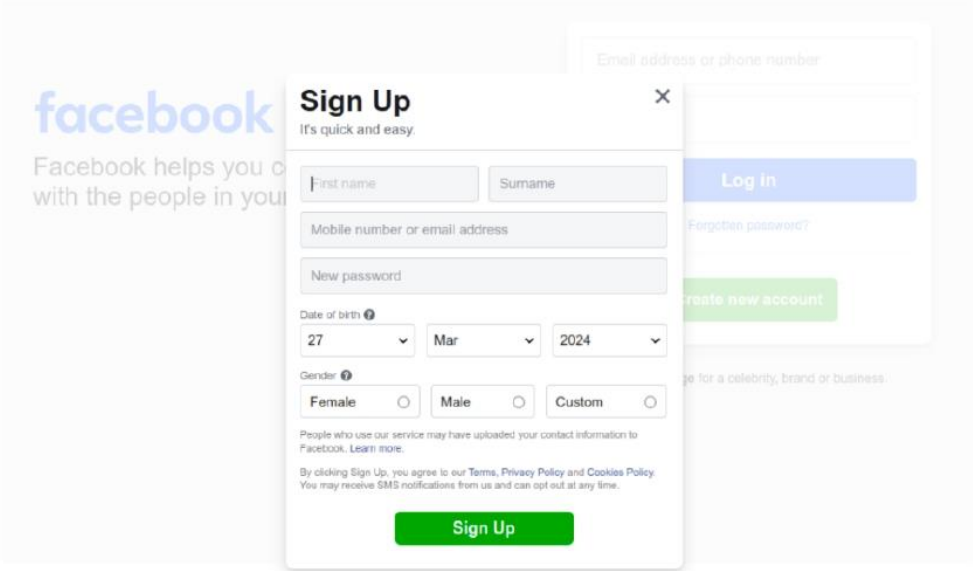


 teqi isə yeni sətirə keçmək üçün istifadə olunur.

HTML formalar

HTML-də forma nə üçündür?

Siz hər hansı bir sayta daxil olduqda orada qeydiyyat üçün məlumatları doldurmusunuz. Hər dəfə həmin veb saytlara daxil olarkən hesabınıza daxil olmaq üçün istifadəçi adı və şifrəni yazırsınız.



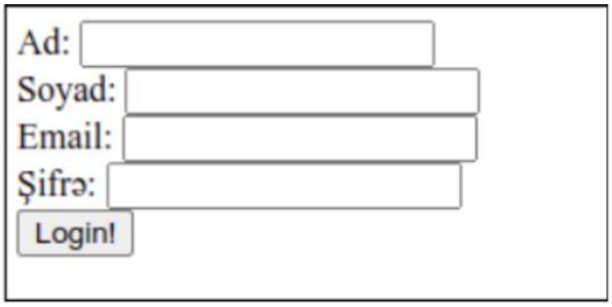
Burada doldurulan məlumat daha sonra serverə göndərilir və orada çalışan proqram təminatı onu emal edir. Bir sözlə HTML dilində formalar istifadəçinin daxil etdiyi məlumatı toplamaq üçün istifadə edilir. Formalar <form> teqi ilə yaradılır. Onun daxilində isə bir sıra digər elementlər yerləşdirilir. Bu teqi konteyner kimi adlandırmaq olar. Nümünə olaraqda, mətn hissələrini, göndərmə düymələrini, radio düymələri və s kimi göstərmək olar.

```html
<form>
    <label for="firstname">Ad: </label>
    <input type="text" name="firstname"  required>
    <br>
    <label for="lastname">Soyad: </label>
    <input type="text" name="lastname"  required>
    <br>
    <label for="email">Email: </label>
    <input type="email" name="email"  required>
    <br>
    <label for="password">Şifrə: </label>
    <input type="password" name="password"  required>
    <br>
    <input type="submit" value="Login!">
</form>
```

Formaların daxilində mətn daxil etmək üçün, “quş qoymaq” üçün, siyahıdan bir elementi seçmək üçün (məsələn, ölkələr siyahısında öz ölkənizi seçmək üçün), tarixi qeyd etmək üçün (məsələn, doğum tarixi), şifrəni daxil etmək üçün və s. elementlər mövcuddur.

İnput elementi

İnput -un bir neçə növü vardır. İnput-da atribut göstərilməyibsə avtomatik olaraq text olaraq qəbul edilir. Belə olduqda o mətni daxil etmək üçün çalışır. Yuxarıdakı kodun nəticəsi aşağıdakı kimi olacaqdır:



İnput elementlərinin növlərinin bütöv siyahısı aşağıdakı kimidir:

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```

Forma doldurulduqdan sonra onun hara yönləndirilməsi üçün formaların “action” attributundan istifadə edilir. Həmçinin formanın məlumatları action-da verilmiş ünvana hansı HTTP metodu ilə göndərilməsi də qeyd edilməlidir. Formanın “method” attributu verilmədikdə onu GET kimi qəbul edir. GET metodu ilə formanın məlumatlarını göndərmək təhlükəlidir. Çünki bu halda formanın keçəcəyi səhifədə bütün input elementlərinin qiymətləri (value xassəsi) ünvanlar barında görünəcəkdir. Ona görə də bu cür hallarda POST, PUT və ya PATCH kimi metodlar istifadə edilir. Məsələn belə bir forma yaradaq:

```html
<form action="https://ozunoyren.com" method="GET">
    <label for="firstname">Ad: </label>
    <input type="text" name="firstname"  required>
    <br>
    <label for="lastname">Soyad: </label>
    <input type="text" name="lastname"  required>
    <br>
    <label for="email">Email: </label>
    <input type="email" name="email"  required>
    <br>
    <label for="password">Şifrə: </label>
    <input type="password" name="password"  required>
    <br>
    <input type="submit" value="Login!">
</form>
```

Bu formada login düyməsini basdıqda növbəti səhifəyə (action-da göstərilən səhifəyə) keçid edəcək və bütün məlumatlar ünvanlar panelində görünəcək:

CSS

CSS nədir?

Cascading Style Sheets (qısa olaraq CSS) - veb səhifənin vizual görünüşündə daha təkmil işlər görmək üçün istifadə edilir. Əgər biz yanlız HTML dən istifadə etsək, veb səhifə çox sadə görünəcək. CSS ilə veb səhifədə bir çox vizual dəyişikliklər etmək mümkündür. Bunlara rəng, fon şəkilləri, fon rəngləri, fərqli ölçülü displeylərdə saytın fərqli formalarda görünməsi, animasiyalar və s. aiddir.

CSS-in HTML-də istifadəsi

CSS həm ayrı bir fayl şəklində, həm HTML faylının head teqləri arasında yeni style teqi açmaqla, həm də HTML teqlərində style atributu vasitəsilə istifadə oluna bilər. Məsələn, belə bir məzmunlu style.css faylı yaradaq:

```css
body {
  background-color: yellow;
}

h1 {
  color: red;
  margin-left: 20px;
}
```

Daha sonra onu HTML faylında istifadə edək:

```html
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

Bunun nəticəsi aşağıdakı kimi olacaq:

CSS faylını analiz edək:

```css
body {
  background-color: #f0f0f0;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Burada ilk sətirdə body sətri HTML-in body elementini 5-ci sətirdəki h1 isə HTMLdəki müvafiq teqi bildirir. Bu o deməkdir ki, onun daxilində, yəni “{“ və “}” şəkilli fiqurlu mötərizə daxilində, yazılmış qaydalar body və h1 teqlərinə şamil olunmalıdır. Burada body və h1 selectorlar adlanır. Bu nümunədə selectorlar ancaq teqlərə aiddir. Selectorlar class və ya id ilə də verilə bilər. Bu o deməkdir ki, ixtiyari bir dəyişən elan edilir və o CSS faylında nöqtə ilə verilir. HTML faylında isə tətbiq olunacaq elementin onu istifadə etməsi üçün class atributunda eyni dəyişənin adı verilir. id ilə olanda isə o # işarəsi ilə yazılır və HTML-də teqin İD attributuna uyğun verilir. Aşağıdakı nümunədə bunu əyani görə bilərsiniz:

```html
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Hello, World!</h1>
    <p class="classNumune">Salamlayırıq hər kəsi. Bu bir paraqrafdır.</p>
    <h2 id="idnumune">İkinci dərəcəli başlıq</h2>
</body>
</html>
```

Uyğun CSS faylı isə belədir:

```css
body {
  background-color: yellow;
}

h1 {
  color: red;
  margin-left: 20px;
}

.classNumune {
    font-size: 20px;
    font-style: italic;
}

#idnumune {
    color: blueviolet;
    background-color: chocolate;
}
```

Nəticə belə olacaqdır:

Geniş istifadə edilən atributlar

Aşağıda qeyd olunan atributlar ən geniş istifadə olunan css atributlarıdır.

atribut	mənası
color	rəng
background-color	arxafon rəng
background-image	arxafon şəkli
display	Konteynerin yerləşmə tərzi {block, inline-block, inline, none}
width	Elementin eni
height	Elementin hündürlüyü
min-width	Minimum en
min-height	Minimum hündürlük
max-width	Maximum en
max-height	Maximum hündürlük
margin	xarici kənarlar (elementin sərhəddindən digər elementə qədər olan məsafə)
padding	daxili kənarlar (elementin sərhəddindən daxildəki elementin sərhədinə qədər olan ara məsafəsi)
border	Sərhəd
border-color	Sərhəd rəngi
border-width	Sərhəd eni
border-style	Sərhəd stili {none, solid, dotted, inset, dashed solid ...}
border-radius	Sərhəd radiusu
font	Şrift imkanları
font-family	Şrift
font-style	Şrift stili {normal, italic, oblique}
font-weight	Şriftin qalınlığı {normal, bold, lighter, bolder}
position	Element üçün istifadə yerləşdirmə , pozisiya növü {static, relative, absolute, fixed, sticky}
z-index	üst üstə düşən elementlərin sırasını təşkil edir. {auto, -1, 0, 1, 2, 3 ...}

Daha geniş siyahını https://developer.mozilla.org/en-US/docs/Web/CSS səhifəsindən əldə edə bilərsiniz.

5 | Mühazirə 5

JavaScript

JavaScript nədir?

JavaScript müasir veb proqramlaşdırmada geniş istifadə olunan proqramlaşdırma dilidir. Əvvəlcə interaktivlik və dinamik məzmun əlavə etməklə veb-səhifələri canlı etmək üçün yaradılmışdır. Hazırda JavaScript bir çox yerlərdə İoT cihazlarında, ağıllı televizorlarda və bir sıra digər sahələrdə istifadə edilir. JavaScript-in veb saytları dinamik etmək üçün imkanlarından danışsaq o, bilavasitə brauzerdə işləyir və skriptlərin yükləndikdən sonra server tərəfində işləməyə ehtiyac olmadan işləməsinə imkan verir.

Brauzerdən başqa, JavaScript, Node.js kimi mühitlər və React Native kimi freymvörklər üçün də əsas dildir. Bu texnologiyaların sayəsində JavaScript mobil proqramlaşdırmada da geniş istifadə edilə bilir. Bundan başqa həm elə React Native, Electron kimi freymvörklər sayəsində JavaScript ilə stolüstü kompüterlər üçün də proqramlar yaratmaq olur. Slack, Visual Studio Code və bir sıra proqram təminatları buna nümunədir.

DOM

Brauzerlərdə JavaScript HTML elementlərini və CSS xassələrini dəyişə bilir. Əslində bu mövzuda məsələ yalnız HTML təqləri ilə bitmir. Brauzer pəncərəsi bir səhifəni vizuallaşdırmaq üçün o səhifənin obyekt modelini qurur. Buna DOM – Document Object Model deyilir. DOM veb səhifələrin proqramlaşdırma interfeysidir. O, səhifəni bu şəkildə təmsil edir ki, həmin interfeysi istifadə edən proqramlar sənədin (saytın) strukturunu, üslubunu və məzmununu dəyişə bilsin.

DOM əslində brauzer pəncərəsindən başlamış səhifədə görünən bütün HTML elementləri, CSS elementlərini özündə cəmləşdirən ağacvari strukturdur. qovşaqlar və obyektlər kimi təmsil edir; bu yolla proqramlaşdırma dilləri səhifə ilə qarşılıqlı əlaqədə ola bilər. Bu, mahiyyətcə JavaScript-ə veb-səhifənin elementlərinə daxil olmaq və manipulyasiya etmək imkanı verir, səhifəni yenidən yükləməyə ehtiyac olmadan dinamik məzmun dəyişikliklərinə imkan verir.

DOM-un mövcudluğu və brauzerlərdə JavaScript istifadə etmə imkanı səhifələri proqramlaşdırmağa imkan verir.

JavaScriptdə "Salam dünya" proqramı.

Brauzer ilə istənilən bir səhifəni açın. DevTools-u aktivləşdirin. Chrome istifadə edirsinizsə, F12 düyməsini basmaqla DevTools açılacaq. Daha sonra Console bölməsinə keçin. Console.log("Salam") yazıb "Enter" düyməsini basın. Bu kod konsola (gördüyünüz ekrana) "Salam" sözünü yazacaq.

Təbrik edirik. JavaScriptdə ilk kodunuzu yazdınız. Burada console DOM-a aid obyektlərdən biridir və brauzerin konsolunu təmsil edir. Və onun log() adlı funksiyası mövcuddur. Bu da həmin konsole hər hansı bir məlumatı loqlaşdırmaq üçün (qeydiyyata almaq üçün) istifadə edilir. Cari nümunədə biz onu sadə şəkildə istifadə etdik və ondan yalnız məlumatın çap olunması üçün istifadə etdik. Biz bu mövzunu daha geniş şəkildə praktikada davam edəcəyik. İndi JavaScriptin HTML səhifəsinə qoşulması və sonra JavaScriptin sintaksisi ilə tanış olaq.

HTML-də JavaScriptin istifadə yolları.

JavaScripti HTML-də istifadə etməyin bir neçə yolu var. Bunlar aşağıdakılardır:

1. HTML-də script teqi yaratmaqla
2. Ayrı fayl yaradıb, JavaScript kodları oraya yazıb, sonra onu HTML sənədinə qoşmaq.
3. Elementlərin xüsusi attributlarını istifadə etməklə.

Hər birinə aid sadə nümunələrə baxaq:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <script>
      console.log("Salam");
    </script>
</body>
</html>

```

Şəkil 5.1: HTML-də script teqi yaratmaq

Bu nümunədə HTML səhifə brauzerdə açılan kimi konsolda ”Salam” sözü çap olunacaq. Eynilə yuxarıda bəhs etdiyimiz nümunədə olduğu kimi.

Ayrı fayl yaradıb, JavaScript kodları oraya yazıb, sonra onu HTML sənədinə qoşmaq.

Eyni şeyi ayrıca fayla da yaza bilərik. ”main.js” adlı aşağıdakı məzmunla fayl yaradaq:

```
console.log("Salam");
```

Sonra onu HTML sənədinə qoşaq:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
  <script src="main.js"></script>
</body>
</html>

```

Sənədi brauzerlə açsanız eyni nəticəni əldə edəcəksiniz. Hər iki yazılış geniş istifadə edi-

lir. Amma bir qayda olaraq HTML sənədini daha anlaşıqlı və oxunabilən saxlamaq üçün JavaScript kodları ayrı faylda saxlamaq daha yaxşıdır. Müasir veb saytlarda minlərlə sətiri olan bir neçə fərqli JavaScript kodları və kitabxanaları istifadə edilir. Onların hər birinin kodlarını HTML sənədinin daxilində yazmaq faylın həcmini də artırar və onun oxunaqlığını aşağı salar.

Elementlərin xüsusi attributlarını istifadə etməklə.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <button onClick="console.log('Salam')">Click me</button>
</body>
</html>
```

Bu HTML faylı brauzer ilə açdıqda bir Button (düymə) elementi olacaq. Onu kliklədikdə konsolda "Salam" sözü çap olunacaq. Burada biz Button teqrinin "onClick" attributunu istifadə etdik. Bu cür attributların siyahılarını internetdə dokumentasiyalardan araşdırmağı sərbəst şəkildə öyrənməlisiniz.

JavaScriptin Sintaksisi

Tədris proqramına görə "Veb təhlükəsizlik" fənni "Proqramlaşdırmanın əsasları" fənnindən sonra gəlir. Müəlliflər nəzərə alıblar ki, oxucular artıq hər hansı bir proqramlaşdırma dilində sərbəst şəkildə proqramlar yazmağı bacarır, demək olar ki, bütün proqramlaşdırma dillərinə xas olan, şərt operatorları, dövr operatorları, riyazi ifadələr, elementar data strukturları, funksiyalar və s. ilə artıq tanışdırlar. Ona görə də o anlayışların detallarına varmadan, bu dərsdə onların JavaScript ilə edilməsini göstərərək JavaScriptin sintaksisini izah edəcəyik.

Əgər sizin üçün bu anlayışlar aydın deyilsə, ilk olaraq "Proqramlaşdırmanın Əsasları" kursunu dinləməniz lazımdır.

Dəyişənlər

Javascript-də dəyişənləri təyin etməyin 3 yolu var bunlar aşağıdakılardır:

1. Let
2. Var
3. Const

var Function scope

var ilə dəyişən elan etdiyimiz zaman yalnız elan edildiyi funksiya daxilində əlçatan olur, funksiya xaricində isə əlçatmaz olur. Əgər dəyişən heç bir funksiya daxilində deyilsə qlobal dəyişən olub window obyektinə mənsub olur.

```
var a = 5;
```

let Block scope

var ilə olduğu kimi let ilə də elan edilmiş dəyişənin dəyərini dəyişdirmək mümkündür, lakin var-dan fərqli olaraq let ilə elan edilmiş dəyişəni eyni blok elementi daxilində yenidən elan etmək olmur.

```
let a = 5;
```

const Block scope

Const ilə təyin olunan dəyişən bir daha dəyəri dəyişə bilməz. Əslində ona dəyişən demək də düzgün deyildir. Bu ədədlər sabit qalır ona görə də işarələnən identifikatora (a sabiti) sabit deyəcəyik. Yazılış qaydaları oxşar olduğundan bir vərdiş olaraq hər kəs sabitlərə də "səhvən" dəyişən deyirlər. Bu deyiliş düzgün olmasa da geniş yayılmışdır və bir mütəxəssis kimi digər mühəndislərin jarqonlarını başa düşmək vacibdir.

```
const a = 5;
```

If operatoru

İf,else,else if- şərt operatorlarıdır. Yazılış qaydası aşağıdakı kimidir:

```
let a = 5;
let b = 10;


if (b === 10 && a === 5) {
   console.log("true");
} else if (b === 10 && a === 3) {
   console.log("a dəyəri düzgün deyil");
} else {
   console.log("error");
}
```

JavaScript dili sintaksis olaraq C dili ailəsinə aiddir. Əgər C, C++, Java, C#, PHP kimi dilləri istifadə etmisinizsə JavaScriptin sintaksisini anlamaq sizin üçün olduqca asan olacaq. Burada 5 və 7-ci sətirlərdə bərabərliyi üç bərabər işarəsi === ilə yoxlamaq diqqətinizi cəlb edə bilər. Məsələ burasındadır ki, JavaScript dinamik tipli dildir. Və digər tərəfdən JavaScriptdə rahatçılıq üçün bir çox tiplər arasında müqayisələr zamanı avtomatik çevrilmələr baş verir.

Məsələn JavaScriptdə belə bir şey yazsanız true qiymətini alacaqsınız:

```
let a = "5";
let b = 5;

a == b;
```

Burada JavaScript onları müqayisə edərkən avtomatik olaraq ortaq bir tipə gətirib müqayisə edir. Ona görə də === bərabərlik işarəsi ilə müqayisə edərək onların identikliyini yoxlamaq lazımdır. Digər proqramlaşdırma dillərindən JavaScript-ə keçid edərkən belə qəribəliklər tez-tez qarşınıza çıxacaq.

Dövrlər

Dövrlər JavaScript-in əsas konstruksiyalarındandır və müəyyən şərtlər altında kod blokunu təkrar-təkrar icra etməyə imkan verir. Onlar təkrarlanan tapşırıqları avtomatlaşdırmaq, massivlər üzərində təkrarlamaq və məlumat kolleksiyalarını idarə etmək üçün xüsusilə faydalıdır. JavaScriptdə dövrlərin yazılış C və ya Java dilinə oxşardır. JavaScriptdə hər biri müxtəlif ssenarilər üçün uyğun olan bir neçə növ dövr vardır. Hər birinə aid sadə nümunələrə baxaq.

For

For dövrü JavaScript-də ən çox yayılmış dövrlərdən biridir. Kod blokunu neçə dəfə yerinə yetirmək istədiyinizi əvvəlcədən bildiyiniz zaman istifadə olunur. O, üç ifadədən ibarətdir: başlanğıc, şərt və son ifadə.

```
for (let i = 0; i < 5; i++) {
    console.log('The number is ' + i);
}
```

Burada i dəyişəninin qiyməti əvvəl 0 olacaq və onun qiyməti 5-dən kiçik olduğu təqdirdə dövrün əsas blokundakı kod icra ediləcək və i-nin qiyməti bir vahid artacaq.

While

Müəyyən edilmiş şərt doğru olduğu müddətdə dövr kod blokunu icra etməyə davam edir. Dövr başlamazdan əvvəl iterasiyaların sayı məlum olmadıqda faydalıdır.

```
let i = 0;
while (i < 5) {
    console.log('The number is ' + i);
    i++;
}
```

Bu iki dövr növündən başqa "do...while", "for...in", "for...of" kimi dövlər də mövcuddur. Onlar barədə ətraflı şəkildə https://www.freecodecamp.org/news/javascript-loops-explained-for-loop-for/ səhifəsindən öyrənə bilərsiniz.

Funksiyalar

JavaScript-də funksiyalar dilin əsas tikinti bloklarından biridir. Onlar eyni kodu yenidən yazmadan bir neçə dəfə icra oluna bilməsi üçün kod blokunu əhatə etmək üçün istifadə olunur. Funksiyalar daha modulyar və saxlanıla bilən kod yazmağa imkan verir. JavaScript-də funksiyaları müəyyən etməyin bir neçə yolu var:

```
function greet() {
    console.log('Hello, World!');
}

greet();

```

Şəkil 5.2: Funksiya açar sözünü istifadə etməklə

```
const greet = function() {
    console.log('Hello, World!');
};
greet();
```

Şəkil 5.3: Funksiyanı ifadə kimi yazmaqla

Ox funksiyaları (arrow functions)

```
const greet = () => {
    console.log('Hello, World!');
};
greet();
```

Digər dillərdə olduğu kimi JavaScriptdə də funksiyalar parametrləri qəbul edə bilər ki, bunlar funksiya çağırıldıqda ona dəyərlər ötürmək üçün istifadə olunur.

```
function add(x, y) {
    return x + y;
}
console.log(add(5, 7));

```

Massivlər

JavaScript-də massivlər digər dillərdəki kimidir. O birdən çox qiyməti bir dəyişəndə saxlamaq üçün istifadə olunur və məlumatları elementin indeksi ilə əldə etməyə imkan verir. Bəzi dillərdən fərqli olaraq JavaScriptdə massivin elementləri fərqli tiplərdə də ola bilər. Bu xüsusiyyət bir sıra dinamik tipli dillərə xarakterikdir.

Aşağıdakı nümunəyə baxaq:

```
let fruits = ['Apple', 'Banana', 'Cherry'];

for (let i=0; i<fruits.length; i++) {
  console.log(fruits[i]);
}
```

Bu kod meyvə adlarından ibarət massiv yaradacaq və dövr vasitəsilə onları konsol-a çap

edəcək.

Yekun

Bu bölmədə JavaScript haqqında qısa şəkildə məlumat verdik və onun sintaksisinə aid sadə nümunələr göstərdik. JavaScriptin imkanları bunlarla bitmir və veb təhlükəsizlik mütəxəssisi kimi siz JavaScript barədə daha geniş biliyə sahib olmalısınız. Bunun üçün fərqli dokumentasiya və ədəbiyyatları istifadə edərək JavaScript biliklərinizi genişləndirməlisiniz. Növbəti mühazirələrdə JavaScript istifadə edərək bir sıra kodlarla tanış olacaqsınız. Orada sizə tanış olmayan sintaksis elementləri və ya proqram kodları ola bilər. Onlar barədə müstəqil araşdırma aparmağı bir mühəndis kimi öyrənməlisiniz.

Regex

RegEx nədir?

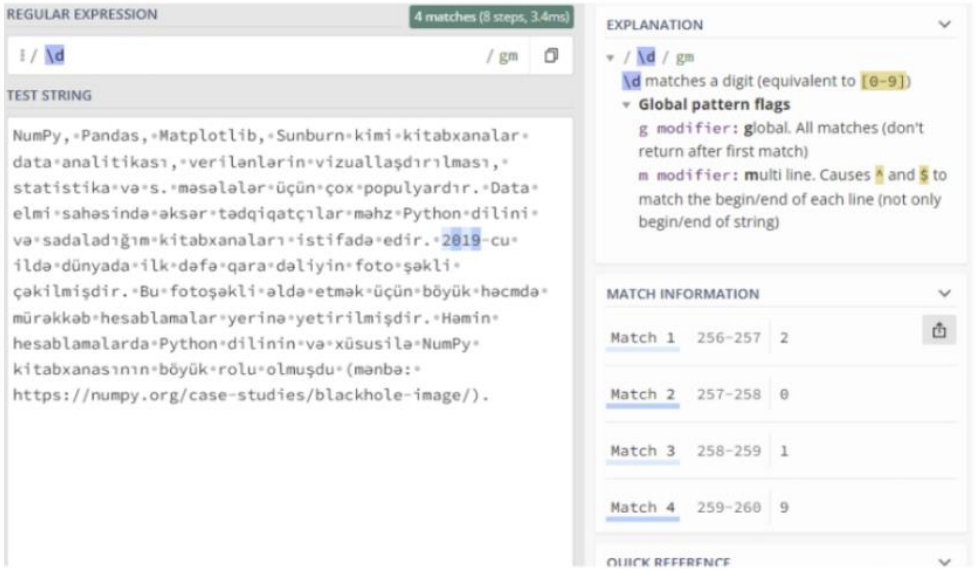
Bu dəfəki mövzumuzda biz proqramlaşdırmada mətnlə işləmək üçün vacib olan bir üsul ilə tanış olacağıq: Requlyar İfadələr və ya qısa şəkildə RegEx. RegEx sətirlərlə işləmək üçün olduqca güclü üsuldur. Mətni dəqiq və səmərəli şəkildə axtarmağa və dəyişdirməyə imkan verir. RegEx mətn nümunələrini təyin etmək üçün ayrıca bir dil kimidir. RegEx ifadələrini yazmaq bəzən tapmacanı həll etmək kimi hiss edilə bilər - nəticədə axtardığınız mətnin qaydalarını müəyyənləşdirirsiniz. Amma bir az əziyyətdən sonra lazımi nəticəsini verir.

RegEx-i öyrənməyin ən yaxşı yollarından biri onunla sınaqlar aparmaqdır. Bunun üçün Regex101 adlı veb saytı istifadə edəcəyik. Regex101.com saytına daxil oluruq. RegEx-i müəyyən qədər öyrənib bu sayt ilə sınaqlar apardıqdan sonra onu JavaScript ilə istifadə etməyi öyrənəcəyik. Regex101 saytı RegEx ifadələrini yazmağa və onları real vaxt rejimində mətnlə tutuşdurmağa imkan verən alətdir.

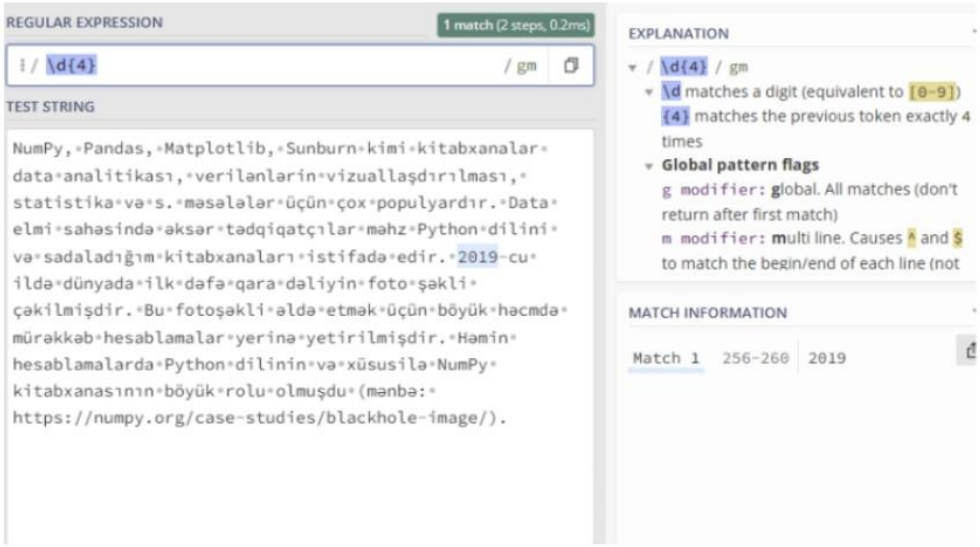
Səhifənin yuxarı hissəsində "Regular Expression" hissəsində mətn RegEx ifadəsini yazacağıq. Siz yazdığınız zaman regex101 sintaksisi də yoxlayır və səhv aşkarlanarsa sizə bildirir. Bu da həm yeni başlayanlar, həm də ekspertlər üçün çox faydalıdır.

Bu xananın altında mətn daxil etmək üçün bölmə var. Üzərində axtarış etmək istədiyiniz mətni buraya yerləşdirmək lazımdır. RegEx ilə tapılan istənilən uyğunluq burada da fərqli rəng ilə işıqlanacaq.

Bu hissədə "MATCH INFORMATION" bölməsini görürsünüz. Tapılman məlumatlar burada görünəcək. Gəlin başlayaq. Belə bir mətni daxil edirəm. Mən indi burada bütün rəqəmləri tapmaq istəyirəm.



"MATCH İNFORMATİON" hissəsində 2, 0, 1, 9 rəqəmlərini verdi və mətndə də o hissəni fərqli rəng ilə işarələdi. Tutalım mən ayrı-ayrı rəqəmləri deyil ili tapmaq istəyirəm. Bilirik ki, hazırda ili göstərən ədəd 4 rəqəmdən ibarətdir. Biz RegEx sintaksisinə görə ifadənin uzunluğunu fiqurlu mötərizənin köməyi ilə təyin edə bilərik. Biz `\d` işarəsinə `{4}` əlavə edərək izah edirik ki, `\d`-yə yəni rəqəmə uyğun olan lakin uzunluğu 4 olan alt-sətirləri tapsın. Və budur 2019 mətnini tapdı.



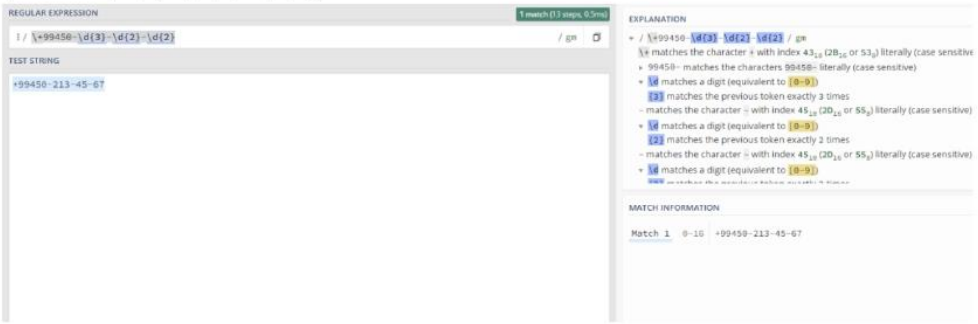
RegEx ilə mətnlərdə məlumatların axtarılması

Təsəvvür edin, istəyirik ki, daxil edilən mətnin Azərbaycan mobil operatorlarına aid telefon nömrəsi olub olmamasını yoxlayaq və bunu etmək üçün regexdən istifadə edək.

Əvvəlcə Azərbaycanda mobil operatorların verdiyi telefon nömrəsinin formatını anlayaq. Azərbaycan telefon nömrələri +99450-213-45-67 kimi formatda olur. Burada +994 ölkə kodu, ardınca iki rəqəmli operator kodu, məsələn, 050, sonra defis işarəsi və yeddi rəqəmli telefon nömrəsi. Burada da ilk üç rəqəm və növbəti hər iki rəqəm cütü defislə ayrılmışdır.

Keçək Regex101 alətinə və başlayaq, biz bilirik ki, Azərbaycan nömrəsi hər zaman +994 ilə başlayır. Ona görə də ölkə kodunu olduğu kimi yazaq. +994. Burada bir məsələ var ki, + işarəsi RegEx dilinin xüsusi ifadəsidir ona görə onu istifadə edə bilmərik. Amma biz onun əvvəlinə \ slash işarəsi əlavə etsək o xüsusi ifadə olmaqdan çıxacaq və adi + işarəsi kimi xarakterizə olunacaq. Gördüyünüz kimi, aşağıda ölkə kodu hissəsi işıqlandır. Artıq biz ifadənin ilk hissəsini tanıya bilirik. Burada 50-ni də əlavə edək. Nömrənin ilk hissəsini tanıdı. Növbəti simvol defisdir, və onu olduğu kimi yazaq.

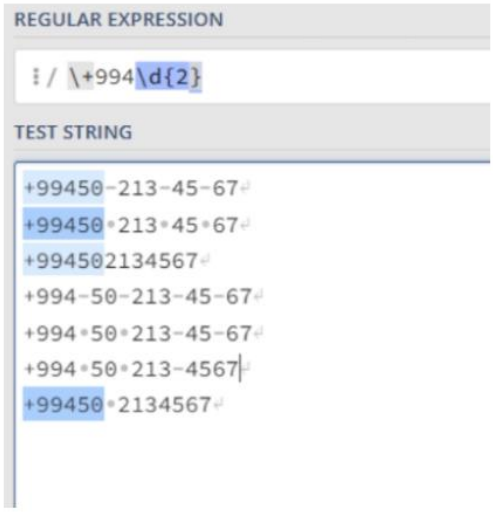
Sonra biz 3 rəqəmli ədədi tanımaq üçün ifadəmisi tamamlamalıyıq. Əvvəlki dərsdən yadınızdadırsa, bunu `\d` işarəsi ilə edirik və sonra rəqəmlərin sayını fiqurlu mötərizədə göstəririk. Budur `\+99450-\d\{3\}`. Eyni qayda ilə işin qalanı asandır. `\+99450-\d{3}-\d{2}-\d{2}` ifadəmiz hazırdır.



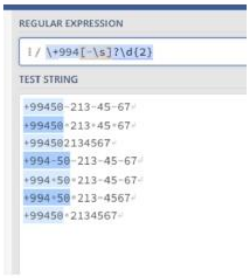
Burada bəzi çatışmazlıqlar var. Çünki, 51, 55, 99, 70, 77 və s. ilə başlayan nörmələr də vardır. Biz onları nəzərə almamışıq. İkincisi Azərbaycanda telefon nömrələri bir qədər fərqli formalarda da yazılır. Məsələn mənim qarşıma bu formatda yazılışlar tez-tez çıxır:

- +99450-213-45-67
- +99450 213 45 67
- +994502134567
- +994-70-213-45-67
- +994 50 213-45-67
- +994 55 213-4567
- +99450 2134567

Biz necə edə bilərik ki, bunların hamısını nəzərə alaq və RegEx ifadəmiz hamısını başa düşüb dəyərləndirə bilsin. Biz operator kodunu ölkə kodundan ayıraq və onu ikirəqəmli ədəd kimi ifadə edək. Nömrələrin bir hissəsini tanıdı. Amma defis ilə olanı tanımadı. Defis yazaq.
+994-{2} İndi isə defis olanı tanıdı o biriləri başa düşmədi.



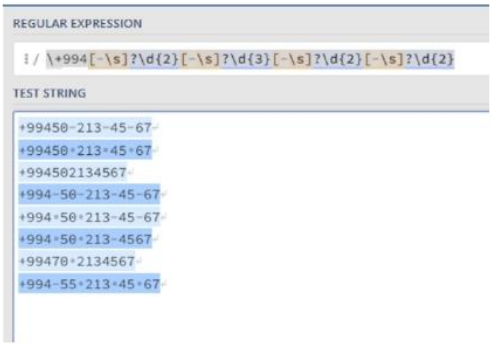
Biz RegEx-ə izah etməliyik ki, burada defis ola da bilər olmaya da bilər. Bunun üçün regexdə ? sual işarəsi istifadə olunur. O hansı simvoldan sonra gəlirsə onun ifadə də həm ola biləcəyini həm də ola bilməyəcəyini izah edir. `\+994-?\d{2}` indi həm defisli həm defissiz olanı tanıdı. Amma boşluq işarəsi olanı tanımadı. Bu dəfə biz elə etməliyik ki, RegEx o hissədə ya defis ya da boşluq işarəsini qəbul etsin. Bunun üçün kvadrat mötərizə istifadə edəcəyik və onun için mümkün simvolları yazacağıq. Boşluq işarəsi `\s` ilə ifadə edilir. `\+994[-\s]?\d{2}`. Biz nömrənin ilk hissəsini tanıyan ifadəni yazdıq.



İndi eyni üsulu istifadə edərək sonrakı hissədə boşluq, defis və ya heç nə, sonra 3 rəqəm, sonra yenə də boşluq, defis və ya heç nə və iki rəqəm və eyni şeyi bir dəfə də təkrarla-yacağıq.

```
\+994[-\s]?\d{2}[-\s]?\d{3}[-\s]?\d{2}[-\s]?\d{2}
```

Digər operatorlara aid nömrələr də yazıb yoxlayaq.



İndi gəlin JavaScript ilə belə bir proqram yazaq. İstifadəçi telefon nömrəsi daxil etməlidir və sizin proqram bu nömrənin doğru olub-olmadığını yoxlamalıdır.

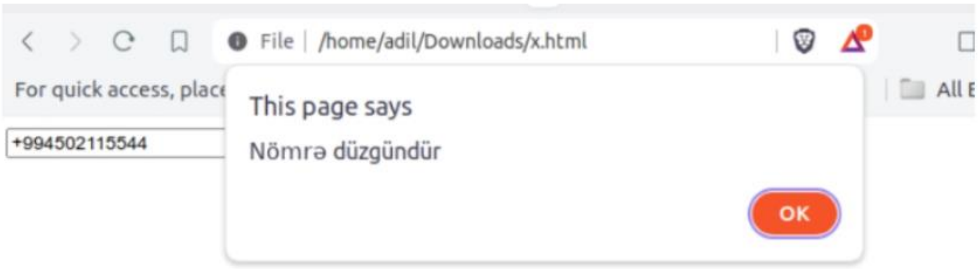
```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <script>
      function yoxla() {
         let textBox = document.getElementById("nomre");
         let nomre = textBox.value;
         let regex = /\+994[-\s]?\d{2}[-\s]?\d{3}[-\s]?\d{2}[-\s]?\d{2}/gm;
         if (nomre.match(regex)) {
            alert("Nömrə düzgündür");
         } else {
            alert("Nömrə düzgün deyil");
         }
      }
   </script>
   <input type="text" id="nomre" />
   <button onClick="yoxla()">Yoxla</button>
</body>
</html>
```

Bu kodda sizə əvvəlki dərslərdən tanış olmayan bir sıra yeniliklər vardır. 11-ci sətirdə

”document.getElementById()” DOM-un document obyektindən getElementById funksiyasını çağırır. Document obyekti carid HTML sənədi bildirən DOM elementidir. Onun getElementById funksiyası isə sənəddə mövcud elementləri əldə etmək üçündür. Arqumentindən həmin elementin İD attributu verilir. 21-ci sətirdə input elementi və onun id attributunu görürsünüz.

getElementById(”nömre”) yazdıqda həmin HTML elementini əldə edib onu textBox adlı dəyişənə mənimsədir. İnput elementlərinin DOM-da ”value” (dəyər/qiymət) attributu vardır ki, daxilində yazılmış mətni bildirir. Biz o attributun qiymətini 12-ci sətirdə ”nomre” adlı başqa bir dəyişənə mənimsətdik. 13-cü sətirdə xüsusi regex sintaksisinə malik ifadəni yazmışıq. Bu sintaksis / ilə başlayı və /gm ilə bitir. Bu iki başlanğıc və son elementin arasındakı, yuxarıda öyrəndiyimiz ifadədir.

14-cü sətirdə match() funksiyası istifadə etmişik. ”match” funksiyası sətir tipli obyektlərə aid funksiyadır və onun arqumentində regex ifadəsi yazılmalıdır. Əgər mətn verilmiş regexə uyğundursa onu regex əsasında ifadə edən yeni bir obyekt qaytarır, uyğun deyilsə null (yəni heç bir şey) qiyməti qaytarır. alert() funksiyası isə brauzer pəncərəsində mesaj qutusu göstərir.



6 | Mühazirə 6

Öyrənmək üçün lazımlı alətlər

Veb təhlükəsizlik ilə bağlı bəzi nəzəri bilikləri öyrəndik, internet protokolları ilə bağlı məlumat əldə etdik, HTML və JavaScript ilə qısa da olsa praktiki təcrübə topladıq. Lakin bunlardan başqa bizim hər gün istifadə edəcəyimiz bir sıra alətləri bilmək vacibdir. Bu alətlər təhlükəsizlik problemlərini tapmağa və həll etməyə, bir sıra analizləri yerinə yetirməyə imkan verir. Bu mühazirədə biz bu vacib vasitələrdən bəzilərinə baxacağıq.

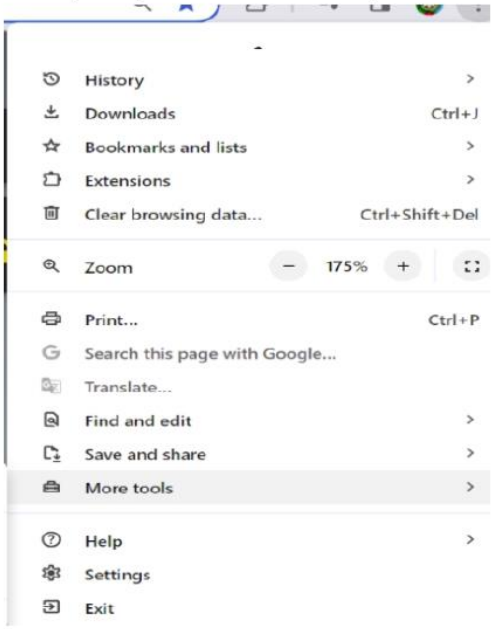
DevTools

DevTools nədir

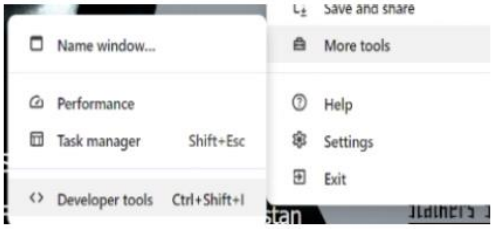
İlk hecası Developer sözündən götürülmüşdür, proqramçılar üçün veb brauzerdə olan bir alətdir. Bu alət istifadəçisinə bir çox imkanlar tanıyır ki, bununla vebsaytları yeniləmək, düzəliş etmək, şəbəkəyə nəzarət etmək, HTML, CSS, JavaScript kodlarına baxmaq redaktə etmək, mobil cihazları simulyasiya etmək mümkün olur. DevTools Google Chrome, Mozilla Firefox, Microsoft Edge, Brave və başqaları daxil olmaqla, əksər müasir veb brauzerlərdə mövcuddur. Bəzi freymvörklərin DevTools üçün genişlənmələri də vardır.

Firefox DevTools/Chrome DevTools

Chrome brauzerində Devtools-a daxil olmaq üçün aşaqdakı addımları izləyə bilərsiniz. Google Chrome-un müvafiq menyusundan More tools (Əlavə Alətlər) bölməsi seçilir.



2.Daha çox alətlər (More tools) bölməsində Developer tools hissəsi seçilir.



Klaviyatura vasitəsilə Ctrl+Shift+ I qısayolları və ya sadəcə F12 istifadə oluna bilər.

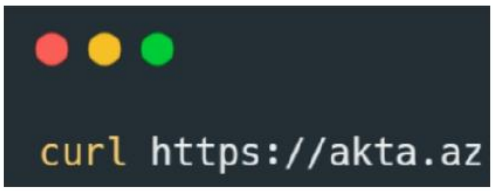
Qeyd: Firefox brauzerində Moretools (Əlavə alətlər), Alətələrin özəlləşdirilməsi bölümündən Web developer tools (veb tərtibatçılar üçün alətlər) hissəsi seçilir. Chrome brauzeri Elements panelində real vaxt rejimində DOM (Sənəd Obyekt Modeli) və CSS üslublarını yoxlamağa və dəyişməyə imkan verir. Siz koda edilən dəyişikliklərin vizual olaraq səhifədə dərhal necə göründüyünə baxa bilərsiniz. Console panelində scriptlərdən qaynaqlanan xətaları və xəbərdarlıqları görmək olar. Xətaların hansı skriptdən qaynaqlandığını araşdırmaq olur. Network panelində isə HTML, CSS, JavaScript və media faylları üçün sorğular və cavablar daxil olmaqla səhifə üçün bütün şəbəkə fəaliyyətini göstərir. Bu, yükləmə performansını təhlil etmək və şəbəkə ilə bağlı problemləri həll etmək üçün çox vacibdir. Application panelində IndexedDB və ya Web SQL verilənlər bazaları, local və sessiya yaddaşı, kukilər və service workers kimi yüklənmiş və səhifə və ya proqram üçün əlçatan olan resursları yoxlamaq üçün alətlər vardır.

DevTools front-end proqramçıların tez-tez istifadə etdiyi alət olsa da, veb təhlükəsizlik mütəxəsisləri üçün də bir sıra analizləri aparmaq üçün lazımdır.

Terminal ilə çalışan alətlər

Curl

Curl HTTP, HTTPS, FTP və s. kimi müxtəlif protokollardan istifadə edərək serverə və ya serverdən məlumat ötürmək üçün istifadə edilən terminal proqramıdır. Curl ilə siz veb səhifənin məzmununu yükləmək kimi tapşırıqları asanlıqla yerinə yetirə bilərsiniz. Məsələn, səhifənin məzmununu kompüterinizə yükləmək üçün aşağıdakı əmri istifadə edə bilərsiniz.



İnternetdən kompüterinizə fayl saxlamaq istəyirsinizsə,

```
curl -o filename.zip https://example.com/filename.zip
```

istifadə edə bilərsiniz, burada -o saxladığınız faylın adını təyin etməyə imkan verir.

Curl-in dokumentasiyasına baxaraq onun digər imkanları barədə də məlumat əldə edə bilərsiniz. Curl ilə siz GET, POST və istənilən HTTP metodu ilə məlumat mübadiləsi edə, başlıqlar, kukilər və s. məlumatları da sorğuya əlavə edə bilərsiniz.

Ümumiyyətlə, curl veb proqramçılar, sistem administratorları və veb serverləri ilə əlaqə saxlamalı və ya məlumat ötürülməsini avtomatlaşdırmağa ehtiyacı olan hər kəs üçün sadə, lakin güclü bir vasitədir. Onun geniş imkanları və müxtəlif protokollar üçün dəstəyi onu şəbəkə ilə bağlı bir çox tapşırıqlar üçün vacib alətə çevirmişdir.

Nslookup

Windows, macOS və Unix/Linux daxil olmaqla, bir çox əməliyyat sistemlərində mövcud olan terminal alətidir. O, domen adı və ya IP ünvanları və ya digər DNS qeydlərini əldə etmək üçün DNS-i sorğulamaq üçün istifadə olunur. Əsasən, nslookup sizə domenin və onunla əlaqəli xidmətlərin təfərrüatlarını öyrənməyə kömək edir.

Nslookup-dan istifadə edərkən siz maraqlandığınız DNS qeydlərinin növünü təyin edə bilərsiniz, məsələn, A, MX, NS və s. Bu, DNS problemlərini həll etmək, domen qeydlərinin düzgün qurulmasını yoxlamaq və ya sadəcə domen adları ilə əlaqəli IP ünvanlarını əldə etmək üçün geniş istifadə edilən alətdir.

Məsələn, belə bir sorğu ilə ozunoyren.com-un MX qeydlərini əldə etmək olar:

```
nslookup -type=MX ozunoyren.com
```

Ping

Ping, İnternet Protokolu (IP) şəbəkəsində hostun əlçatanlığını yoxlamaq üçün istifadə olunan proqram təminatıdır. O, təyinat kompüterinə göndərilən mesajlar üçün gediş-gəliş vaxtını ölçür. Ping əməliyyatı sadədir; o, ICMP protokolu ilə “əks-səda” sorğu paketlərini hədəf hosta göndərir və ICMP əks-səda cavablarını dinləyir. Bununla da o, şəbəkə bağlantısının keyfiyyətini qiymətləndirmək üçün gediş-gəliş vaxtını (RTT) və paket itkisi faizini hesablayır.

```
ping example.com
```

Ping əmrinin nəticəsi aşağıdakılardır:

- Hədəf hostun IP ünvanı.
- Göndərilən ICMP paketlərinin ölçüsü.
- Hər bir paket üçün gediş-gəliş vaxtı, adətən millisaniyələrlə.
- Göndərilən, qəbul edilmiş, itirilmiş paketlərin xülasəsi və itki faizi.

Gediş-gəliş vaxtışəbəkə bağlantısının sürətini və sabitliyini qiymətləndirmək üçün çox vacibdir. Yüksək gediş-gəliş vaxtı dəyərləri və ya əhəmiyyətli paket itkisi sıxlıq, marşrutlaşdırma problemləri və ya hostun özü ilə bağlı problemlər kimi müxtəlif məsələlərlə bağlı ola biləcək şəbəkə problemlərinə işarə edir. Ping şəbəkə bağlantısını və performansını tez yoxlamaq üçün faydalı alət olsa da, onun məhdudiyyətləri də mövcuddur. Bəzi qurğular və ya şəbəkələr ICMP trafikini bloklaya bilər ki, bu da pingin cavab almasına mane olur və beləliklə, host işləmirmiş kimi görünür. Üstəlik, ping şəbəkə problemlərinin təbiəti haqqında ətraflı diaqnostika təmin etmir; o, yalnız hostun əlçatan olub olmadığını göstərir və əsas gecikmə məlumatını təmin edir.

Tracet

Tracert və ya traceroute mənbədən təyinat yerinə bir IP şəbəkəsi üzrə paketlərin marşrutlaşdırma yollarını diaqnostika etmək üçün istifadə edilən proqramdır. Bu, müəyyən bir IP ünvanı və ya domenə çatmaq üçün şəbəkə üzrə paketlərin keçdiyi yolu müəyyən etməyə kömək edir. Paketlərin keçdiyi marşrutlaşdırıcıların (hops) siyahısını göstərməklə, gecikmələr və ya nasazlıqların baş verdiyi yerlər də daxil olmaqla, şəbəkənin performansı haqqında məlumat verir.

Sniffer

Snifferlər şəbəkə təhlili, təhlükəsizlik və problemlərin aradan qaldırılmasında istifadə olunan alətlərdir. Onlar şəbəkə vasitəsilə axan məlumat paketlərini ələ keçirmə əsasında işləyir, istifadəçilərə bu paketlərin məzmununu yoxlamaq, deşifrə etmək və təhlil etmək imkanı verir. Snifferləri şəbəkə trafikinin nümunələrini anlamaq, şəbəkə problemlərini diaqnostika etmək, təhlükəsizlik pozuntularını müəyyən etmək və məlumatların düzgün ötürülməsini təmin etmək üçündür.

Snifferlər proqram təminatı və ya cihaz şəklində ola bilər. Proqrama əsaslanan snifferlər kompüterdə işləyir və trafiki ələ keçirmək üçün qeyri-adi rejimdə şəbəkə interfeysi kartından (NIC) istifadə edir. Avadanlıq snifferləri şəbəkəyə qoşulmuş müstəqil cihazlardır və şəbəkə performansına təsir etmədən trafiki ələ keçirmək üçün xüsusi olaraq hazırlanmışdır.

Snifferlər tərəfindən tutulan məlumatlara mənbə və təyinat IP ünvanları, port nömrələri, protokol məlumatları və ötürülən faktiki məlumatlar daxil ola bilər. Snifferlər şəbəkə administratorları və təhlükəsizlik mütəxəssisləri üçün güclü alətlər olsa da, onlar həm

də güclü şəbəkə təhlükəsizliyi tədbirlərinə ehtiyacı vurğulayaraq, parollar və şəxsi məlumatlar kimi həssas məlumatları ələ keçirmək üçün zərərli şəxslər tərəfindən istifadə edilə bilər.

Populyar snifferlərə Wireshark, tcpdump və Ettercap daxildir ki, onların hər biri paket tutmaq və təhlil etmək üçün müxtəlif funksiyalara malikdir. Snifferlərdən etik istifadə şəbəkə sahiblərindən icazə tələb edir, çünki icazəsiz sniffer istifadəsi məxfilik qanunlarını və etik standartları poza bilər.

Wireshark

Wireshark məşhur pulsuz və açıq mənbəli paket analizatorudur (sniffer) və şəbəkə problemlərinin həlli, təhlili, proqram təminatı və protokolların yaradılması üçün sınaqlar və təhsil məqsədləri üçün geniş istifadə edilən proqram təminatıdır.

Wireshark-ın əsas funksionallığı, real vaxt rejimində şəbəkədən keçən məlumatları tutmaq və onu istifadəçiyə göstərməkdən ibarətdir. Wireshark-ın imkanları TCP, UDP, HTTP, FTP və DNS kimi tanış olanlar da daxil olmaqla 2000-dən çox şəbəkə protokolunun təhlilini əhatə edir. O, Ethernet, Bluetooth və Wireless (IEEE.802.11) kimi geniş şəbəkə növlərində canlı şəbəkə trafikini tutmağa imkan verir. Həmçinin topladığı məlumatı fayl kimi saxlaya bilir, başqaları tərəfindən toplanmış məlumatları (trafikdən toplanmış məlumatları) təhlil üçün açıb oxuya bilir. Bu, Wireshark-ı həm real vaxt rejimində problemlərin təhlili, həm də hadisədən sonrakı təhlil üçün bir vasitə halına gətirir.

Proqramı https://www.wireshark.org/ saytından yükləyə bilərsiniz.

API alətləri

Postman və Hoppscotch kimi API alətləri veb-proqramçıların tez-tez istifadə etdikləri alətlərdən olsalar da, veb təhlükəsizlik mütəxəssislərinin alətlər qutusunda da vacib alətlərdəndir. Bu alətlər müasir veb proqramların mühüm komponentləri olan API-lərin dizaynını, sınaqdan keçirilməsini və idarə olunmasını asanlaşdırır. API-lərə sorğu göndərmək və cavablara baxmaq üçün istifadəçi interfeysi təmin etməklə, bu alətlər təhlükəsizlik mütəxəssislərinə zəiflikləri müəyyən etməyə, təhlükəsizlik tədbirlərini sınaqdan keçirməyə və API-lərin təhlükəsizlik üzrə ən yaxşı təcrübələrə (best practices) uyğun olmasını təmin etməyə kömək edir.

Bir qədər əvvəl CURL haqqında danışmışdıq. Bütün sorğuları CURL sorğular kimi yazmaq bir qədər vaxt alır. Ona görə də Postman və ya Hoppscotch kimi alətlər daha səmərəlidir və vaxta qənaət etməyə imkan verir.

Postman istifadəçilərə API-ləri yaratmağa (APİ spesifikasiyalarını), paylaşmağa, sınaqdan keçirməyə və sənədləşdirməyə imkan verən məşhur API alətidir. O, REST, SOAP və GraphQL daxil olmaqla müxtəlif növ sorğuları dəstəkləyir.

OpenAPI

Əvvəllər Swagger Spesifikasiyası kimi tanınan OpenAPI Spesifikasiyası (OAS), RESTful API-lər üçün standart interfeysdir. O, həm insanlara, həm də kompüterlərə xidmətin imkanlarını onun mənbə koduna daxil olmadan və ya hər hansı sənədi görmədən başa düşməyə imkan verir. Spesifikasiyaya aşağıdakılar haqqında məlumatlar daxildir:

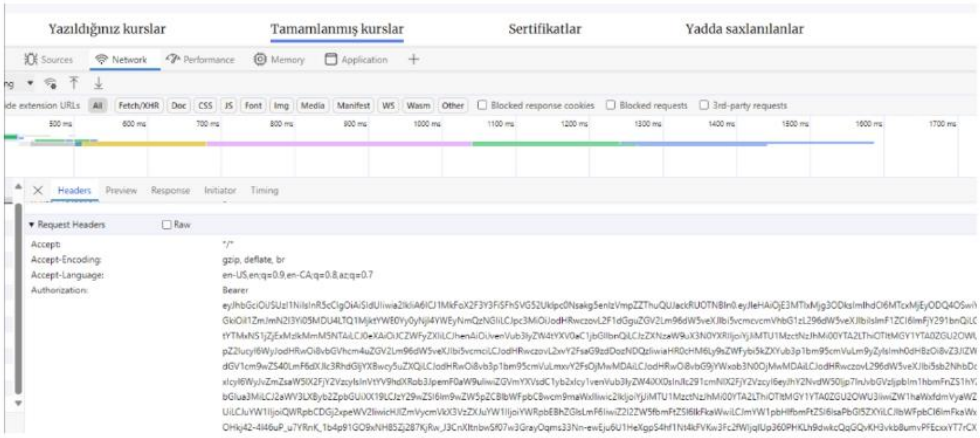
- Mövcud son nöqtələr (/istifadəçilər, /yazılar və s.) və hər bir son nöqtə üzrə əməliyyatlar (GET, POST, DELETE və s.)
- Əməliyyat parametrləri Hər əməliyyat üçün giriş və çıxış
- Validasiya üsulları
- Əlaqə məlumatları, lisenziya, istifadə şərtləri və digər müvafiq məlumatlar

OpenAPI açıq standart olduğundan, o, geniş çeşidli alətlər və xidmətlər tərəfindən dəstəklənir.

7 | Mühazirə 7

Sessiyalar

Təsəvvür edin ki, xoşladığınız geyim mağazanızdan onlayn alış-veriş edirsiniz. Sayta daxil olursunuz, müxtəlif kateqoriyaları nəzərdən keçirirsiniz, alış-veriş səbətinizə bir cüt ayaqqabı və köynək əlavə edirsiniz, lakin sonra alışınızı tamamlamazdan əvvəl gedirsiniz çay içməyə. Bir saatdan sonra sayta qayıtdığınız zaman görürsünüz ki, alış-veriş səbətinizdə hələ də əvvəllər əlavə etdiyiniz ayaqqabı və köynək var və siz hələ də sistemdə aktivsiniz. Brauzer sizi saytda necə aktiv saxlayır? Nə üçün hesabı tərk etməmisiniz? Server ümumiyyətlə haradan bilir ki, bu bir saat əvvəlki istifadəçidir? Ümumiyyətlə sistemə daxil olduqdan sonra, bir səhifədən digərinə keçəndə yeni səhifə istifadəçi haqqında məlumatı haradan bilir? Detallara baxmadan əvvəl, belə bir araşdırma edək. İstənilən bir veb sayta keçib, hesabınıza daxil olun. Məsələn, https://learn.ozunoyren.com . Hesabınıza daxil olduqdan sonra brauzerinizdə DevTools-u aktivləşdirib Networks bölməsinə keçin. Daha sonra istənilən bir səhifəyə keçid edin. DevTools-un Networks bölməsində açdığınız səhifəyə uyğun HTTP sorğularını görəcəksiniz. O sorğunun “Request Headers” bölməsinə nəzər yetirin. Təqribən belə bir şey olacaq:



Burada “Authorization” başlığı (header) cari istifadəçi haqqında məlumatları özündə ehtiva edir. Təbii ki, bu məlumatlar şifrələnmiş şəkildədir və onu ancaq server açа biləcək. Çünki o məlumatı ilkin olaraq generasiya edən də elə serverdir.

İndi isə gəlin, bu prosesin necə işlədiyinə dair detallarla tanış olaq.

Vebsayta daxil olduğunuz zaman server ziyarətiniz üçün unikal sessiya ID-si yaradır. Bu sessiya ID brauzerinizə geri göndərilir və adətən kukidə (bəzən local storage adlı yerdə) saxlanılır. Brauzer növbəti dəfə eyni sayt daxilində başqa bir səhifəyə keçid edəndə məzmunla birlikdə (məsələn, alış-veriş səbətinin məzmunu) bu sessiya ID-sinə aid məlumatları da serverə göndərir.

Fasilə verdiyiniz zaman brauzer seansınız aktiv olaraq qalır. Brauzerinizdə saxlanılan sessiya ID-si təmin edir ki, siz qayıdıb yeni sorğular göndərdiyiniz zaman server hələ də sizi tanısın. Geri qayıtdıqdan sonra, siz sorğu göndərən kimi (məsələn, alış-veriş səbəti səhifəsinə keçmək) brauzeriniz saxlanılan eyni sessiya ID-ni serverə göndərir. Server seans məlumatlarınızı axtarmaq üçün bu ID-dən istifadə edir.

Kukilər

Bayaqkı eksperimentimizin davamını yerinə yetirək. Kompüterinizdə brauzeri tamamilə bağlayın və ya kompüteri yenidən başladın (restart edin). Öz e-poçtunuza, sosial şəbəkə hesabınıza və ya hər hansı bir hesabınız olan səhifəni açdıqda yenidən login olmağa ehtiyac qalmır. Bu necə baş verir? Brauzer hardan bilir ki, siz həmin istifadəçisiniz?

Bunun üçün brauzerlər kukilər adlı texnologiyanı istifadə edir. Sadə şəkildə desək kukilər (cookie) brauzerin yaddaşında saxlanılan kiçik ölçülü məlumatlardır. Hər bir sayt özünə aid kukini sizin brauzerinizdə saxlaya bilər.

Eyni domenə (veb sayta) etdiyiniz hər sorğu ilə kukilər serverə geri göndərilir. DevTools-un network bölməsinə bir daha nəzər yetirməklə buna şahidlik edə bilərsiniz. Bu mexanizm kompüterinizi yenidən başlatdıqdan sonra belə serverə sizi yadda saxlamağa və daxil olmanıza imkan verir. İlkin olaraq vebsayta daxil olduqda, server brauzerinizə sessiya identifikatoru (sessiya ID) ilə kuki göndərir. Bu sessiya identifikatoru giriş statusunuzun və digər sessiya məlumatlarınızın saxlandığı serverdəki sessiyanızı müəyyən edən unikal sətirdir.

Brauzer bu kukini öz yaddaşında saxlayır. Hər dəfə həmin sayt daxilində istənilən sorğunu göndərdikdə sorğu ilə birlikdə kukiləri də serverə göndərir. Beləliklə, kompüterinizi yenidən başlatdıqdan sonra geri qayıtdığınız zaman brauzer sessiya ID-si olan kukini yenidən serverə göndərir. Server sessiya identifikatorunu oxuyur, sessiya məlumatlarınızı alır (əvvəllər daxil olduğunuzu təsdiqləyir) və daxil olma vəziyyətinizi saxlayır.

Server kuki üçün sessiya İD-sini yaradandan sonra öz verilənlər bazasında saxlayır. Və müəyyən müddət orada qalır. Onun orada nə qədər qalması serverin konfiqurasiyalarından asılıdır.

Local storage (Lokal yaddaş)

“Local storage” veb tətbiqləri üçün istifadəçinin brauzerində məlumatları saxlamaq metodlarından biridir. Web Storage API-nin bir hissəsidir. “Local storage”-in meydana çıxmasından əvvəl kukilər istifadə olunurdu. Kukilərin yaddaş məhdudiyyətləri (adətən 4KB-a qədər) var və müasir texnologiyalar daha çox yaddaşa ethiyac duymağa başlamışdılar. Digər tərəfdən kukilər təbiətinə görə belə çalışır ki, onlar hər dəfə serverə ötürülür. Veb tətbiqlər isə heç də bütün yaddaş məlumatlarını serverə ötürmək məcburiyyətində deyil. Bu cür ehtiyaclar Web Storage API-ın meydana gəlməsi ilə

nəticələndi.

Web Storage API əslində həm də HTML5-in tərkib hissəsidir. HTML5-dən əvvəl veb proqramçıları məlumatları brauzerdə saxlamaq üçün kukilər və ya əlavə plaginlər istifadə etməklə müxtəlif üsullardan istifadə etməli idilər. Bu üsullar təkcə imkan baxımından məhdud deyil, həm də təhlükəsizlik baxımından səmərəli deyildi. Web Storage API daha böyük həcmdə məlumatların yerli olaraq saxlanması üçün daha möhkəm və təhlükəsiz yol təqdim etməklə bu problemləri həll etmək üçün daha geniş HTML5 spesifikasiyasının bir hissəsi kimi ortaya çıxdı. Web Storage API-nin tərkib hissəsi olan "Local storage" hər bir domen üçün key-value cütü formatında 5MB-a qədər (və ya bəzi brauzerlərdə daha çox) məlumat saxlamağa imkan verir. Kukilərdən fərqli olaraq, bu məlumatlar hər HTTP sorğusu ilə serverə göndərilmir.

Hazırda "Local storage" aşağıdakılar üçün geniş istifadə olunmaqdadır: Tətbiqin yüklənmə vaxtını yaxşılaşdırmaq və serverə gedəcək sorğuların sayını azaltmaq üçün proqram məlumatlarının keşləşdirilməsi. İstifadəçi parametrlərini saxlamaq. veb tətbiqlərin müvəqqəti məlumatlarının saxlanması.

Kukilərlə müqayisədə aşağıdakı üstünlüklərə malikdir: Yaddaş: Kukilərlə müqayisədə əhəmiyyətli dərəcədə daha çox yer təklif edir (5MB və ya daha çox) və daha mürəkkəb strukturlu məlumatların saxlanmasına imkan verir. Performans: Məlumatları hər sorğu ilə serverə göndərilmədiyi üçün lazımsız şəbəkə trafikini və yükləmə vaxtını azaldır. Davamlılıq: Lokal yaddaşda saxlanılan məlumatlar skript vasitəsilə və ya istifadəçi tərəfindən aydın şəkildə təmizlənənə qədər qalır və uzunmüddətli məlumatların saxlanması üçün onu daha etibarlıdır. Kukilərdə olan məlumatı isə server "cookie" başlıqları göndərməklə dəyişə bilər.

Lokal yaddaş müştəri tərəfində böyük həcmdə məlumatların saxlanması üçün səmərəli üsul olsa da, qeyd etmək vacibdir ki, o, sessiyanın idarə edilməsi və ya autentifikasiya məqsədləri üçün kukiləri birbaşa əvəz etmir. Kukilər istifadəçi seanslarını saxlamaq üçün serverə sessiya identifikatorları və ya tokenlər göndərmək üçün hələ də lazım olur. Kukilər serverə hər sorğu ilə birlikdə avtomatik göndərilir. Lokal yaddaşdakı məlumatlar isə serverə avtomatik şəkildə heç vaxt göndərilmir.

Lokal yaddaşın üstünlüklərini nəzərə alaraq bir çox proqramçılar kuki əvəzinə lokal yaddaşdan da istifadə edirlər. Məsələn, sessiya ilə əlaqəli məlumatları JavaScript ilə lokal yaddaşa yazırlar və proqramçılar kodu elə yazır ki, veb tətbiq hər dəfə sorğu göndərərkən lokal yaddaşdan lazım olan məlumatları serverə göndərsinlər. Eynilə kukinin avtomatik etdiyinə oxşar şəkildə. Belə yanaşmalarla proqramçılar bəzən kukiləri lokal yaddaşla əvəz edirlər.

Web Storage API (https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API)

Sessiyaların təhlükəsizliyi

Same origin policy

Vahid Mənbə Siyasəti (SOP) potensial zərərli veb-saytların müxtəlif mənşəli məlumatlara daxil olmasının və ya dəyişdirilməsinin qarşısını almaq üçün veb brauzerlər tərəfindən həyata keçirilən kritik təhlükəsizlik mexanizmidir. SPO mənbə URL-in protokolu (http və ya https), host (domen) və portu (məsələn, 80, 443, 8080 və s.) ilə müəyyən edilir. SOP-a əsasən, bir mənşədən veb-səhifədə işləyən skriptə eyni mənşədən məlumat əldə etməyə icazə verilir, lakin digərindən deyil.

Məsələn, sizin saytınız ozunoyren.com ünvanında yerləşir. Saytınızda bir iframe var və o akta.az saytına yönləndirilib. Yəni ozunoyren.com səhifəsinin içində həm də akta.az səhifəsi açılır. SOP-a görə ozunoyren.com səhifəsindəki heç bir JavaScript akta.az saytındakı elementlərə, nə də akta.az saytı ozunoyren.com saytına çıxış əldə edə bilməz.

Vahid Mənbə Siyasətinin əsas məqsədi istifadəçi məlumatlarını qorumaq və zərərli skriptlərin digər veb-saytlardan məlumatlara sərbəst daxil olmasına icazə verildiyi təqdirdə baş verə biləcək təhlükəsizlik pozuntularının qarşısını almaqdır. Məsələn, SOP olmadan bir veb-saytdakı skript potensial olaraq istifadəçinin daxil olduğu başqa saytdan məlumatları oxuya, dəyişdirə və ya ötürə bilər ki, bu da saytlararası skript (XSS) hücumlarına, məlumat oğurluğuna və ya sessiyanın oğurlanmasına səbəb olur.

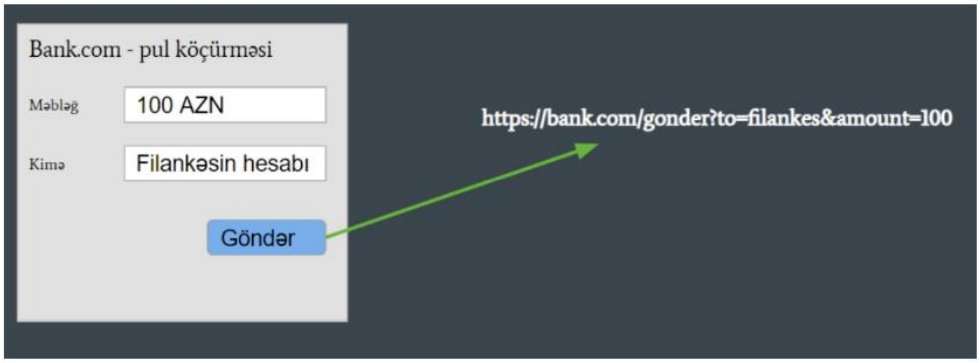
Təsəvvür edin ki, SOP yoxdur. Siz gmail.com saytında login olmusunuz. Brauzeriniz gmail.com-a aid sessiyanı öz kukilərində və ya lokal yaddaşında saxlamışdır. Hər hansı zərərverici bir sayt bunu ehtimal edərək öz səhifəsində xüsusi bir JavaScript yazmış və bu JavaScript səhifəki iframe-in daxilindəki başqa bir səhifnin kuki və lokal yaddaş məlumatlarını götürə bilir. Əgər belə ssenari mümkün olsa idi, onda sizin gmailinizi ələ keçirmək üçün dələduzlar öz saytlarında gmailə yönləndirilmiş bir iframe istifadə etmələri bəs idi. Bu çox böyük bir təhlükəsizlik problemi olardı.

Vahid Mənbə Siyasətinin məqsədi elə bu ssenarinin qarşısını almaqla istidaçinin məlumatlarını qorumaqdır.

Bəzi hallarda hər iki saytın müəllifi özümüz olarkən, bir səhifədən iframe ilə olan digər səhifəyə həqiqətən də məlumat göndərmək istəyirik. CORS (Cross-Origin Resource Sharing) və postMessage kimi mexanizmlər SOP üçün bu hallarda istisnalara imkan verir. CORS serverlərə öz aktivlərinə kimin daxil ola biləcəyini bəyan etmək üçün bir üsul edir və postMessage funksiyası ilə brauzerin pəncərələrin (tabları və ya frameləri) arasında etibarlı əlaqə yaratmaq mümkün olur. Bu cür məlumat mübadiləsi yuxarıdakı təhlükəsizlik riskini artırmır. Bu halda hər iki tərəf nəyi və necə mübadilə edəcəyini bildiyi üçün təhlükəsizlik riski minimuma enmiş olur.

Cross-Site Request Forgery

Fərz edək ki, siz öz onlayn bank hesabınıza daxil olmusunuz. Oraya daxil olduqdan sonra dostunuza pul göndərirsiniz. Bunu etdikdə bankın veb saytında belə bir sorğu formalaşır:



Bu sorğunun "to" hissəsində pulu alacaq adamın hesab nömrəsi, "amount" hissəsində isə məbləğ bildirilib.

İndi təsəvvür edin ki, bir dələduz veb sayt hazırlayıb. Və orada kiçik və görünməz bir şəkil yerləşdirib:

Şəkilin ünvanı eynilə sizin bankınızın sorğu URL-idir. Və burada o "to" hissəsində öz hesabının nömrəsini yazıb. Əgər siz həmin sayta daxil olsanız sizin hesabınızdan 100 AZN dələduzun hesabına köçəcək.

Əlbəttə bu cür sorğuları əslində GET üsulu ilə deyil POST üsulu ilə etmək lazımdır. Amma o zaman dələduz img teqi ilə yox iframe teqi ilə və bir neçə sətir JavaScript kodu ilə eyni şeyi yenə də edə bilərdi.

Bu cür hücumlara CSRF (Cross-Site Request Forgery – Saytlararası Sorğu Saxtakarlığı) deyilir. Bu hücum istifadəçinin autentifikasiya məlumatlarını onların xəbəri və ya razılığı olmadan ələ keçirmək və onun adından sorğular göndərmək üçün hədəflənib.

CSRF hücumlarının qarşısını almaq üçün aşağıdakı taktikalardan istifadə etmək olar:

Anti-CSRF Tokenləri istifadə etməklə sorğunun tətbiqin öz formasından qaynaqlanmasına əmin olmaq. Burada səhifəyə ilk olaraq serverdən gələn token əlavə edilir. Sorğu serverə getikdə onun həqiqiliyi yoxlanılır və o token ləğv edilir. Bir token ikinci dəfə istifadə edilə bilmir. Əgər dələduz istifadəçidən hər hansı bir sorğunun formasını tokenlə birlikdə ələ keçiribsə, o vaxtı keçmiş tokeni əldə etmiş olacaq. Vaxtı keçmiş tokenlə gələn sorğunu isə server qəbul etməyəcək.

CSRF-in qarşısının alınma üsullarından biri də SameSite Cookie Atributunu istifadə etməkdir. Kukilərdə SameSite atributundan istifadə brauzerə kuki-ni yalnız hədəf saytla eyni domendən gələn sorğularda göndərməyi tapşırır və kukinin üçüncü tərəf saytları tərəfindən istənilən sorğularla birlikdə göndərilməsinin qarşısını alır.

Başqa bir taktika isə HTTP sorğusunun “Referer” başlığının yoxlanılmasıdır. Bu üsul bəzi hallarda səmərəli olsa da , təhlküəsizlik baxımından tövsiyə edilmir. Çünki Referrer başlığını “yaratmaq” elə də çətin iş deyil. Dələduzlar bunu ustalıqla bacar bilərlər.

8 | Mühazirə 8

Kod inyeksiyası (Code İnjection) nədir?

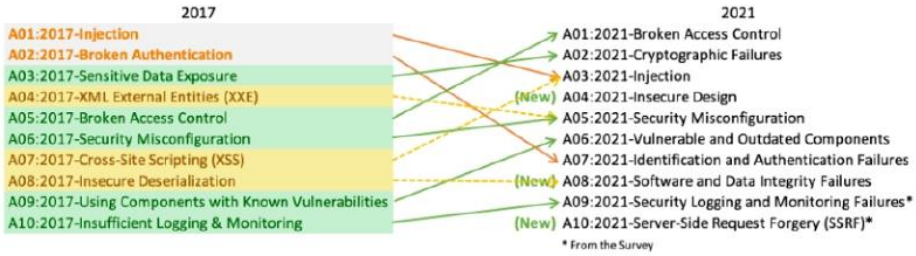
Kod inyeksiyası təcavüzkarın zərərli kodu sizin proqram təminatına yeritmək və o kodu sizin sistemdə icra etmək üçün veb proqramdakı zəifliklərdən istifadə etdiyi geniş kiber-hücumlar sinfini təmsil edir. Bu üsul proqram sistemlərinin təhlükəsizliyini, bütövlüyünü və əlçatanlığını poza bilər. Belə ki, icazəsiz girişə, məlumatların pozulmasına və bir sıra digər təhlükəsizlik problemlərinə səbəb ola bilər.

Kod inyeksiyası ilə yeridilmiş kod müxtəlif növlərdə ola bilər. Məsələn, SQL, JavaScript, HTML və ya PHP və ya Python kimi serverdə icra edilən skript dilləri, hədəf tətbiqin zəifliklərindən asılı olaraq, digər dillər və s. Kod inyeksiyası sanki zərərli "maddəni" (yəni, kodu) sizin sistemə iynə ilə yeridilməsidir. Buna görə də məcazi mənada bu hücum növünə "kod inyeksiyası" deyilir.

Bu mühazirədə biz Cross-Site Scripting (XSS) və Əməliyyat sistemlərinə kod iyeksiyası ilə olan hücumlardan danışacağıq. Həmçinin SQL İnjection hücumları barədə qısa söhbət açacağıq. SQL injection haqqında daha geniş şəkildə növbəti mühazirələrdə öyrənəcəyik.

Kod inyeksiyasını başa düşmək həm proqramçılar, həm də təhlükəsizlik mütəxəssisləri üçün çox vacibdir.

Qısa olaraq desək, etibarsız, şübhəli, yoxlanılmamış parametr gözlənilmədən koda çevrilirsə, buna kod inyeksiyası deyəcəyik. Kod inyeksiyası OWASP-ın tərtib etdiyi demək olar ki, bütün hücumlar reytinqində yer alır. Bu o deməkdir ki, kod inyeksiyası hər bir zaman diqqət olunmalı hücum növüdür.



Kod inyeksiyasının qarşısını alma üsulları kifayət qədər sadə olsa da proqramçılar tərəfindən edilən çox kiçik bir diqqətsizlik bütün sistemi risk altında qoya bilir. 2014-cü ilə SQL inyeksiyası vasitəsilə hakerlər Tesla şirkətinin serverindən istifadəçi məlumatlarını oğurlamışdılar. 2018-ci ildə eyni boşluq Cisco şirkətində Cisco Prime License Manager sistemində aşkarlanmışdır. Hakerlər özlərinə qeyri-qanuni yolla rəsmi lisenziyalar əldə etmişdilər. 2015-ci ilə eBay veb-saytında XSS boşluğu aşkar olunmuşdu. Hakerlər URL paremetrlərini dəyişməklə istifadəçi məlumatlarını oğurlamağa müvəffəq olmuşdular. Facebook şirkəti ən güclü proqramçıları işə götürməklə məşhur olsa da 2011-ci ildə Facebookda aşkarlanmış XSS boşluğu xeyli istifadəçiyə ziyan vurmuşdur.

İndi gəlin kod inyeksiyası növlərinin bəziləri haqqında daha ətraflı öyrənək.

Command injection

Hücum

İlk olaraq "command injection" – əmr inyeksiyası haqqında danışaq. Əmr inyeksiyası dedikdə proqram təminatına hansısa parametr vasitəsilə əməliyyat sistemində yerinə yetirilən əmrlərin yeridilməsi başa düşülür. Bu tip hücumda hücumda istifadəçi (dələduz) proqram vasitəsilə əməliyyat sistemində ixtiyari əmrləri yerinə yetirə bilir. Bu zəiflik ona görə mövcud ola bilir ki, proqram hər hansı bir parametri istifadəçidən qəbul edir və onu düzgün şəkildə yoxlamadan sistemdə icra edir.

Belə bir nümunəyə baxaq. Fayl axtarış əməliyyatını yerinə yetirmək üçün istifadəçi daxiletməsindən istifadə edən Node.js proqram kodunu nəzərdən keçirək. Proqram istifadəçi girişindən bir fayl adını alır və onu faylı əldə edən shell əmrinə ötürür. Müvafiq giriş yoxlaması olmadan, bu əməliyyat təcavüzkar tərəfindən ixtiyari əmrləri yerinə yetirmək üçün istifadə edilə bilər.

Aşağıdakı koda nəzər yetirək:

```
1  const express = require('express');
2  const { exec } = require('child_process');
3
4  const app = express();
5
6  app.get('/file', (req, res) => {
7    const userFile = req.query.filename;
8    // Vulnerable command execution
9    exec(`cat ${userFile}`, (error, stdout, stderr) => {
10     if (error) {
11       return res.status(500).send(`Error: ${error.message}`);
12     }
13     if (stderr) {
14       return res.status(500).send(`Error: ${stderr}`);
15     }
16     return res.send(stdout);
17   });
18 });
19 app.listen(3000, () =>
20   console.log('Server running on port 3000');
21 });
22
```

Bu kodun 6-cı sətrində onun /file ünvanlı GET sorğusu gözlədiyini görürük. Məsələn, veb sayt http://numune.az ünvanındadırsa http://numune.az/file URL-inə daxil olduqda bu kod işə düşəcək. Kodun 7-ci sətrində GET sorğusunun filename parametri qəbul edilir və userFile dəyişəninə mənimsədilir. Məsələn, http://numune.az/file?filename=salam daxil ediləndə userFile dəyişəninin qiyməti "salam" olacaq.

Kodun 9-cu sətrincə exec funksiyasının köməyi ilə onun birinci arqumentindəki Linux əmri yerinə yetirilir. Belə olduqda gördüyünüz kimi cat əmri icra edilir. Yəni, Linux sistemi cat salam komandası icra edilir. Linuxda cat əmri mətn faylının məzmununa baxmaq üçündür. Mətn faylının məzmunu stdout dəyişəninə yazılır və HTTP sorğusuna cavab olaraq geri göndərilir. Beləliklə yuxarıdakı kod filename parametri ilə gələn faylın məzmunu oxuyub cavab olaraq məzmunu müştəriyə göndərir. İlk baxışdan hər şey düzgün görünür. Əslində bu cür kod düzgün işləyir, lakin haker gözü ilə baxdıqda burada böyük bir problem olduğunu görərik. Təsəvvür edin ki, mən sorğunu bu şəkildə dəyişirəm: `http://numune.az/file?filename=salam; ls`

Belə olduqda yekunda icra ediləcək əmr bu olacaq: 'cat salam; ls'. Nəticə salam faylının məzmunu və həmin fayl olan direktoriyadakı bütün faylların siyahısı olacaq (ls əmri direktoriyadakı faylların siyahısını göstərir). Burada ; işarəsi ilə bir neçə əmri ayrı-ayrılıqda icra etmək mümkündür. Bu o deməkdir ki, biz sistemə nəzərdə tutulmayan başqa əmrləri də yeridə bilirik. Əgər biz məsələn 'rm -rf *' yazsaq həmin direktoriyadakı faylları silə bilərik. Və yaxud sistemdən şifrələri, SSL sertifikatlarını, konfiqurasiyaları əldə edə bilərik. Sistemə zərərli proqramlar yükləyə bilərik. Və bir sıra digər zərərli işləri yerinə yetirə bilərik.

“Command injection” hücumundan müdafiə

Bu cür hücumlardan müdafiə üçün bir sıra yanaşmalar mövcuddur. İlk məsələ odur ki, parametrləri yoxlamadan onu sistemdə icra etmək olmaz. İstifadəçidən gələn giriş məlumatlarına heç vaxt inanmaq olmaz. Ona görə də bu parametrlər ciddi şəkildə yoxlanılmalıdır. Cari nümunədə yaxşı olar ki, NodeJS-in faylları oxumaq üçün mövcud olan müvafiq funksiyalarından istifadə edilsin. Bir-başa sistemə əmr göndərmək heç də yaxşı üsul deyil. Heç bir halda mümkün olmadıqda dəyişənləri ciddi şəkildə yoxlamaq lazımdır. Məsələn, yuxarıdakı nümunədə faylın adı ola biləcək simvolların olub-olmadığını yoxlamaq yaxşı olardı. Söhbət konkret Linux və Unix tipli əməliyyat sistemlərindən gedirsə bu `https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html#Quoting` resursdan istifadə edərək hansı işarələrə diqqət etməli olduğunuza qərar verə bilərsiniz.

Proqramçı gözü ilə yuxarıdakı tədbir əslində pis deyil. Amma proqramçılar dedik ki, diqqətsizlik edə bilirlər. Bu hətta böyük şirkətlərdə də baş verir. Ona görə “Ehtiyat yaxşı şeydir” prinsipinə dayanaraq əlavə üsullar da yerinə yetirilsə yaxşı olar. Məsələn, veb saytı icra edən istifadəçini ayırmaq olar və o istifadəçiyə yalnız lazım olan məhdud səlahiyyətləri vermək olar. Belə olduqda həmin sistemə verilən əmr əməliyyat sisteminin digər sahələrinə zərər vurmayacaqdır.

LDAP Injection

LDAP Injection istifadəçi girişi əsasında LDAP (Lightweight Directory Access Protocol) sorğularını yaradan koda zəifliklərdən istifadə edərək müdaxilə edəməklə hücum növüdür. LDAP direktoriya xidmətlərindən istifadə etmək üçün bir protokoldur. Məsələn, çox geniş yayılmış direktoriya servisinə Microsoft Active Directory-ni göstərək olar. Bu gün bir çox təşkilatlarda Microsoft Active Directory şəbəkə daxilində istifadəçilər, qruplar və digər resurslar haqqında məlumatı saxlamaq və əldə etmək üçün istifadə edilir. Həmin directoriya ilə inteqrasiya olunmuş sistemləri istifadəçilərin məlumatlarına girişi LDAP vasitəsilə yerinə yetirir.

Məsələn, bank işçisi öz işini evdən yerinə yetirə bilmək üçün sistemə daxil olub öz istifadəçi adı və şifrəsini yazaraq sistemdən istifadə edir. Burada sistem LDAP sorğusu ilə istifadəçinin sistemə girişini təmin edir. Yəni istifadəçinin verdiyi parametrlər haqqında bir LDAP sorğusu formalaşır və yerinə yetirilir. Veb tətbiq LDAP sorğularına daxil edilmiş məlumatı düzgün yoxlaya bilmədikdə LDAP inyeksiya zəifliyi yaranır. Bu cür zəiflik olduqda dələduzlar direktoriyaya LDAP vasitəsilə sorğular göndərə bilər və sistemə zərər vura bilərlər.

Hücum

LDAP injection hücumu o zaman baş verir ki, təcavüzkar istifadəçi daxiletməsi əsasında qurulan sorğuya zərərli LDAP sorğusunu daxil edə və ya ”inyeksiya” edə bilsin. Tətbiq is-

tifadəçi daxiletməsini birbaşa qəbul edərsə və lazımi təmizləməni yerinə yetirməzsə onu LDAP sorğusuna birləşdirərsə inyeksiya hücumu baş tutar. Beləliklə təcavüzkar tərəfindən daxil edilən zərərli LDAP ifadələri sorğunu həssas məlumatları aşkar etmək kimi gözlənilməz hərəkətlər etmək üçün dəyişə bilər.

LDAP əsasında istifadəçiləri autentifikasiya edən veb tətbiqini nəzərdən keçirək. Tətbiq istifadəçidən istifadəçi adı və şifrəni alır və sonra LDAP kataloqu ilə yoxlamaq üçün LDAP sorğusu qurur. Qurulan sorğu aşağıdakı kimidir:

```
(&(uid=[username])(userPassword=[password]))
```

Bu sorğuda username və password istifadəçidən gələn məlumatlardır və sorğuya, fərz edək ki, yoxlanmadan daxil edilir. Bu sorğuya görə uid (yəni istifadəçi identifikatoru) və userPassword sistem dəyişənlərinə verilən məlumatlar mənimsədilir və onların hər ikisinin eyni zamanda doğru olduğu şərt qarşısındakı & ilə yoxlanılır. Tətbiq istifadəçi girişini lazımi qaydada təmizləmirsə, təcavüzkar LDAP sorğusunu dəyişdirmək üçün istifadəçi adı və ya parol kimi xüsusi hazırlanmış başqa "məlumat" daxil edə bilər. Məsələn, istifadəçi adını bu cür yaza bilər:

```
*)(uid=*))(|(uid=*
```

Belə olduqda yekun LDAP sorğusu aşağıdakı kimi olacaq:

```
(&(uid=*)(uid=*))(|(uid=*)(userPassword=[password]))
```

Bu o deməkdir ki, bu dəfə & ilə (uid=*) və (uid=*) sorğuları yoxlanılır. Bu o deməkdir ki, uid istənilən olan hal və iki dəfə eyni şeyi təkrarlayıb onları məntiqi "və" ilə birləşdirir. Daha sonra "|" məntiqi "və ya" ilə (uid=*) və (userPassword=[password]) şərtlərini yoxlayır. Bu şərtlərdən ikisindən biri doğru olarsa ümumi ifadə doğru olmuş olur. İlk şərt isə uid-in istənilən qiymətində doğru olur.

Yekun ifadə "doğru" ilə nəticələnir və haker bu minvalla sistemə daxil ola bilir.

Qarşısının alınması

LDAP inyeksiya hücumlarından qorunmaq üçün veb proqramlar aşağıdakı təhlükəsizlik tədbirlərini həyata keçirməlidir:

Girişin təmizlənməsi: Bütün istifadəçi daxiletmələrini ciddi qaydalar dəstinə (məsələn, uzunluq, format, icazə verilən simvollar) uyğun olaraq yoxlanmalı və potensial təhlükəli simvollar silinməli və ya kodlaşdırılmalıdır. Bunun üçün OWASP-ın tövsiyələr sənədindən istifadə etmək olar:
https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html.

Parametrləşdirilmiş Sorğuların İstifadəsi:
Bəzi LDAP kitabxanaları parametrləşdirilmiş sorğuları dəstəkləyir, burada parametrlər

sətirlərin sadə birləşməsi deyil, təmizlənməklə xüsusi şəkildə sorğu formalaşdırma baş verir.

İmtiyazları məhdudlaşdırmaq:
Tətbiqlər LDAP inyeksiya hücumunun potensial təsirini məhdudlaşdıran ən az imtiyazlara malik hesablar altında işə salınmalıdır. Buraya LDAP sorğuları vasitəsilə hansı məlumatların əldə edilə və ya dəyişdirilə biləcəyini məhdudlaşdırmaq daxildir. Belə olduqda sistemdə hətta boşluq olsa da haker çox da zərər vura bilməyəcək.

Cross-Site Scripting (XSS)

Aşağıdakı koda nəzər yetirək:

```
app.get('/', (req, res) => {
    const q = req.query.q
    res.send(`
        <h1>
            Axtardığınız sorğu: ${q}
        </h1>
    `)
})
```

Burada sistem GET metodu ilə gələn sorğuda q parametrini h1 teqləri arasında çıxışa verir. Yəni brauzer ilə bu sayta daxil olduqda, ünvan http://numune.az/?q=salam olarsa, nəticə olaraq veb saytda “Axtardığınız sorğu: salam” mətnini görəcəksiniz.

Kodun 2-ci sətirinə diqqət etsəniz, q parametrini elə olduğu kimi q dəyişəninə mənimsədir. Heç bir yoxlama aparılmır. Belə olduqda təsəvvür edin ki, ünvan belə yazılır: http://numune.az/?q=Salam. Sayta bu cür daxil olduqda o teqini də səhifədə istifadə edəcək. Bu o deməkdir ki, q parametrindən gələn istənilən HTML kodu səhifəyə yeridilə bilir. HTML kodunun yerinə script teqinin arasında JavaScript kodu da icra edilə bilər. Bu isə ciddi bir təhlükəsizlik boşluğudur.

Bu cür hücumlara XSS (Cross-site scripting) hücumu deyirlər. XSS istifadəçi daxiletmələrini lazımi səviyyədə yoxlamayan veb proqramlara təsir edən geniş yayılmış təhlükəsizlik zəifliyidir. XSS hücumları təcavüzkarlara digər istifadəçilərin baxacağı məzmuna zərərli skriptlər yeritməyə imkan verir. XSS hücumunun mahiyyəti təcavüzkarın digər istifadəçilər tərəfindən baxılan veb səhifəyə və ya proqram çıxışına zərərli JavaScript, HTML və ya digər icra edilə bilən kodu daxil etməsini nəzərdə tutur. Bu digər istifadəçilər zərərli məzmuna baxdıqda, daxil edilmiş kod onların brauzerində icra olunur və potensial olaraq məlumatların oğurlanmasına, sessiyanın oğurlanmasına, veb səhifələrin pozulmasına və digər zərərli fəaliyyətlərə gətirib çıxarır.

Qarşısının alınması

Digər inyeksiyas hücumları kimi XSS-in də qarşısının alınmasında əsas yeri daxiletmənin yoxlanması və icazə verilməyən işarələrin ləğv edilməsi tutur. Bəzi kitabxanalar (məsələn, React) bu işi avtomatik görür. React komponentlərinin məzmununa HTML mətnini verəndə onu HTML kimi yox adi mətn kimi qəbul edir. Belə yanaşma təhlükəsizlik riskini xeyli aşağı salır.

Digər üsul isə Content Security Policy (CSP) adlı texnologiyanı istifadə etməkdir. Hazırda bütün brauzerlər tərəfindən dəstəklənir. CSP etibarlı veb səhifə kontekstində zərərli məzmunun icrası nəticəsində yaranan XSS, kliklәmə və digər kod inyeksiya hücumlarının qarşısını almaq üçün təqdim edilmiş təhlükəsizlik standartıdır. CSP veb server administratorlarına hansı mənbələrdən gələn kodlara icazə verilməsini təmin edir. Bu siyasət brauzerə hansı resursların yüklənməsinə və icrasına icazə verilməli olduğuna qərar verməyə kömək edir və bununla da zərərli kodun icrası riskini azaldır.

CSP veb serverin HTTP cavab başlığı vasitəsilə həyata keçirilir. Bu başlıq brauzerlərin səhifə məzmununu yükləyərkən riayət etməli olduğu siyasət və ya qaydalar toplusunu ehtiva edir. Bu qaydalar hansı domenlərin JavaScript, CSS, şəkillər, şriftlər və digər resurslara xidmət göstərməsinə icazə verildiyini, həmçinin daxili skriptləri məhdudlaşdıra bilir.

Məsələn, Content-Security-Policy: script-src 'self' https://apis.example.com;

Bu siyasət brauzerə yalnız səhifənin özündən və https://apis.example.com saytından olan skriptləri icra etməyə imkan verir. Hər hansı digər skriptlər, o cümlədən daxili skriptlər və müxtəlif domenlərdən olan skriptlər isə bloklayır.

XSS hücumları məlumatları əldə edib harasa göndərir və məlumatları hakerlər bu şəkildə oğurlaya bilirlər. Bunun üçün onlar məlumatı əldə etdikdən sonra səhifəni harasa yönləndirməli, yaxud başqa səhifədən olan bir skripti icra etməli ya da analoji yollarla (məsələn, img teqini istifadə etməklə) manipulyasiyalar edirlər. CSP isə inanılmamış resursları bloklamağa kömək edir. Bu isə nəticə etibarı ilə XSS hücumunun qarşısını almağa müvəffəq olur.

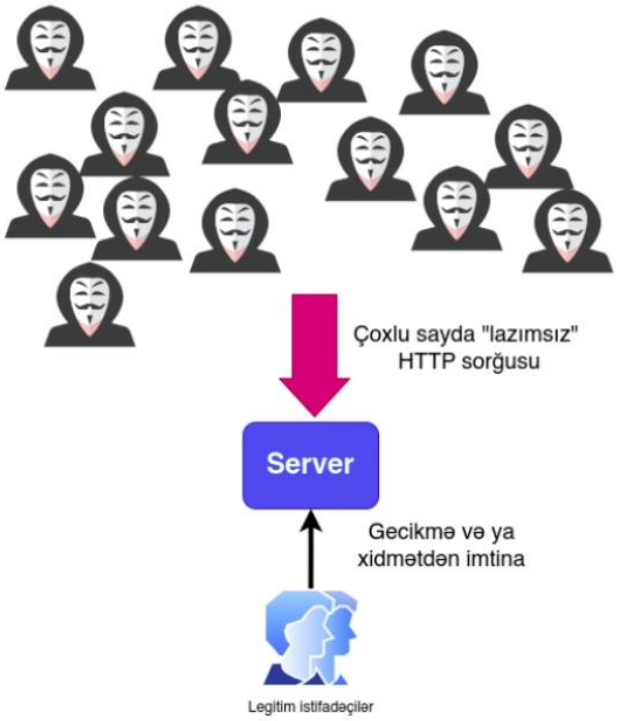
9 | Mühazirə 9

DoS (Denial of Service)

DoS (Denial of Service) nədir?

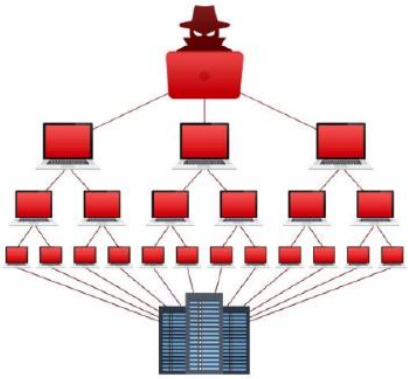
DoS (Denial of Service) hədəf sistemə və ya şəbəkəyə böyük həcmdə məlumat trafiki göndərməklə girişin qarşısını almaq və ya xidməti yararsız hala gətirmək məqsədi daşıyan kiberhücum növüdür. Hər bir sistemin idarə edə biləcəyi şəbəkə trafikinin həcmi var. Sistemin bu resursları təcavüzkarlar tərəfindən həddindən artıq yükləndikdə, sistem xidmətləri yavaşlayır və hətta bu hücumlar nəticəsində sistemin təqdim etdiyi xidmətlər tamamilə çökür. Bu cür hücumlara DoS, ingilis dilində Denial of Service və ya Xidmətdən İmtina deyilir.

DDos (Distributed Denial of Service) hücum isə bir mənbədən deyil, bir çox müxtəlif mənbələrdən başladıqda baş verir. DDos hücumlarını həyata keçirmək üçün adətən zombi adlanan cihazlardan ibarət botnetlərdən istifadə edilir. Bu zombi cihazları internet hakerləri tərəfindən ələ keçirilən elektron cihazlardır və təcavüzkarların məqsədləri üçün istifadə olunur. DDos hücumları arzulanan məqsədə çatmaqda Dos hücumlarından daha uğurlu olur. Birdən çox mənbədən hədəfə aparıldığı üçün əsas mənbəni aşkar etmək çətinləşir.



Bu tip hücum müxtəlif məqsədlər üçün həyata keçirilə bilər, məsələn, şirkətləri, dövlət qurumlarını, internet saytlarını və ya onlayn xidmət təminatçılarını hədəfə ala bilər.

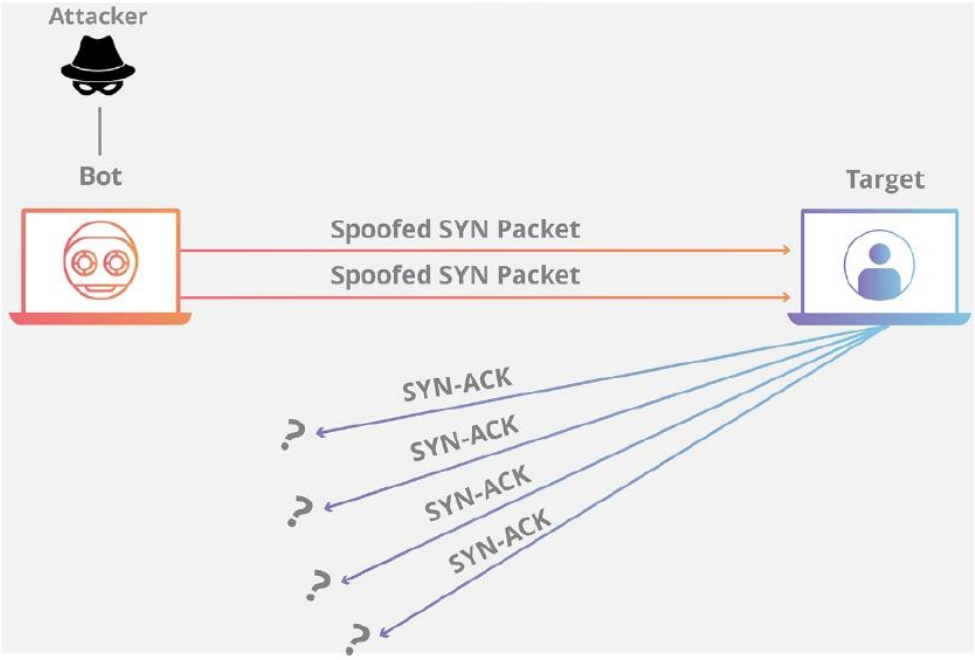
DDoS hücumları sistemi həddən artıq yükləmək üçün normal istifadəçiləri və ya qanuni trafiki manipulyasiya edərək xidmətdən imtinaya səbəb olur. DDoS hücumu təcavüzkarın nəzarəti altında olan bir sıra kompüterlər (botnet adlanır) və ya cihazlardan istifadə etməklə həyata keçirilir. Bu botnet təcavüzkarların sındırma və ya zərərli proqram vasitəsilə ələ keçirdikləri cihazlar toplusudur. Bu botnetlərdən istifadə edərək, təcavüzkarlar hücum trafikini hədəfə yönəldir, hədəf sistemdə həddən artıq yüklənməyə və normal trafiki emal etməkdə çətinlik yaradır.



DoS və DDoS hücumlarının növləri

Volume Based DDoS: Sadəcə olaraq serverin gücü çata biləcəyindən daha çox həcmdə məlumat göndərilməsidir.
Protocol Based DDoS: Bu hücum, OSI modelinin 3-cü (Network) və 4-cü (Transport) layları üzrə zəifliklərdən istifadə etməklə yerinə yetirilir.
Application Layer DDoS: Hücum OSI modelinin 7-ci qatı olan tətbiq səviyyəsində xidmətlərin zəifliklərindən istifadə etməklə həyata keçirilir.
HTTP Flood: Bu hücumda daim hədəf səhifəyə almaq və ya göndərmək sorğuları göndərməklə sistemi yükləyir.
UDP Flood: Hücum UDP protokolundan istifadə etməklə həyata keçirilir. Təcavüzkar tərəfindən çoxlu sayda UDP paketləri kompüterin portlarına göndərilir. Hücumun hədəfi olan kompüter portun istifadə vəziyyətini yoxlayır və istifadə edilmədikdə ICMP paketi ilə cavab verir. Çox sayda UDP paketinə cavab olaraq çoxlu sayda ICMP paketləri göndərilir. Beləliklə, sistem iflic vəziyyətətə düşür.
ICMP Flood: ICMP protokolu ICMP sorğu paketlərini hədəf sistemə göndərir və digər sistemdən cavab gözləyir. Bu şəkildə çoxlu sayda sorğuya cavab verməyə çalışan sistem iflic olur.
Ping of Death: Hədəf sistemə həcmcə böyük ICMP sorğu paketi göndərməklə həyata keçirilir.
Syn Flood: SYN "daşqın" hücumları TCP bağlantısının "əl sıxma" prosesindən istifadə etməklə işləyir. Normal şəraitdə TCP bağlantısı əlaqə yaratmaq üçün üç fərqli proses yerinə yetirir. Əvvəlcə müştəri əlaqəni başlamaq üçün serverə SYN paketi göndərir. Bundan sonra server rabitəni təsdiqləmək üçün həmin ilkin paketə SYN/ACK paketi ilə cavab verir. Nəhayət, müştəri paketin serverdən alınmasını təsdiqləmək üçün ACK paketini qaytarır. Bu paketin göndərilməsi və qəbulu ardıcıllığını tamamladıqdan sonra TCP bağlantısı açılır və məlumat göndərə və qəbul edə bilir. Syn Flood üsulu ilə DoS hücumu etmək üçün təcavüzkar elə edir ki, ilkin SYN paketi alındıqdan sonra server bir və ya bir neçə SYN/ACK paketi ilə cavab verəcək və əl sıxmada son addımı gözləyəcək. Bu necə

işləyir. Təcavüzkar hədəflənmiş serverə yüksək həcmli SYN paketləri göndərir, paketdə mənbə olaraq saxta IP ünvanlarını göstərir. Bundan sonra server qoşulma sorğularının hər birinə cavab verir və cavabı qəbul etməyə hazır portu açıq buraxır. Server heç vaxt gəlməyən son ACK paketini gözləyərkən, təcavüzkar daha çox SYN paketi göndərməyə davam edir. Hər yeni SYN paketinin gəlişi serverin müəyyən müddət ərzində müvəqqəti olaraq yeni açıq port bağlantısını saxlamasına səbəb olur və bütün mövcud portlar istifadə edildikdən sonra server normal fəaliyyət göstərə bilmir.



Mənbə: https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/
Bunlardan başqa TearDrop, Smurf və s. kimi DoS hücum növləri də vardır.

Botnetlər

Botnetlər botlar və ya zombilər kimi tanınan, botmaster adlandırılan təcavüzkar tərəfindən idarə olunan kompüter şəbəkələridir. Bu şəbəkələr internetə qoşulmuş kompüter və cihazlarda zəifliklərdən istifadə edən zərərli proqramların yayılması ilə yaradılır. Bir dəfə yoluxmuş bu maşınları sahiblərinin xəbəri olmadan uzaqdan idarə etmək olar.

Botmasterlər bu kompüterləri bir qayda olaraq pis niyyətlər üçün istifadə edirlər. Bunlara aşağıdakıları nümunə göstərmək olar:
DDoS hücumları
Spamların göndərilməsi
Məlumat oğurluğu

Kriptovalyuta mədənçiliyi
Zərərli proqramların yayılması və s.

Kompüterlər botnetlərə adətən zərərverici proqramların yoluxması nəticəsində qoşulur. Bunun qarşısını almaq üçün kibergigiyena qaydalarına əməl etmək lazımdır. Məsələn, zərərli proqram təminatını aşkar etmək və silmək üçün müasir antivirus və anti-malware həllərindən istifadə etmək bu addımlardan biridir. Bundan başqa zərərli proqram tərəfindən istifadə edilə bilən zəifliklər aradan qaldırmaq üçün əməliyyat sistemlərini və proqram təminatını mütəmadi olaraq yeniləmək lazımdır. Təbii ki, həm də təhlükəsizlik haqqında hər zaman məlumatlı olmaq, şübhəli saytlara daxil olmamaq, qeyri-qanuni proqram təminatlarını istifadə etməmək lazımdır.

DDoS-dan müdafiə

DDoS-dan müdafiə mümkündürmü?

DDoS hücumları idarə edə bildiklərindən daha çox trafikə malik veb-saytları, xidmətləri və ya şəbəkələrə dayanmadan hücum etmək və onları əlçatmaz etmək üçün nəzərdə tutulub. DDoS hücumlarının qarşısının alınması onların paylanmış təbiəti və yarada biləcəyi böyük trafik həcminə görə çox çətindir. Nəzəri olaraq qeyri-mümkündür. Lakin, təşkilatlar DDoS hücumlarının riskini və təsirini azaltmaq üçün bir neçə strategiya həyata keçirə bilər.

Məsələn, şəbəkə infrastrukturunu birdən çox data mərkəzləri və coğrafi məkanlar arasında paylaşdırmaq olar. Belə olduqda hücumlar fərqli data mərkəzlərə yönələcək və hücumun hər data mərkəzə düşən həcmini azaldacaq. Yaxud hücum yalnız bir şəbəkəyə yönlənibsə digər şəbəkədəki resurslara təsiri olmayacaq.

Həmçinin proqram təminatlarında gecikmələrə səbəb olan kodların optimallaşdırması da DDoS riskini bir qədər azalda bilər. Məsələn, təsəvvür edin ki, sizin saytınıza gələn sorğu nəticəsində proqramınız verilənlər bazasından məlumatları əldə edir və mürəkkəb SQL sorğuları vasitəsilə emal edib müştəriyə göstərir. Bu sorğu hər dəfə eyni məlumatı qaytarırsa o zaman onu keşləmək olar ki, verilənlər bazasına hər dəfə sorğu getməsin. Belə olduqda sistem verilənlər bazası ilə hər dəfə əlaqə saxlamayacaq və SQL sorğusunu icra etmək üçün vaxt sərf etməyəcək. Beləliklə sayt daha cəld çalışacaq. Bu da o deməkdir ki, serverin TCP bağlantılar daha az müddətə açıq qalacaqlarına görə sərbəst bağlantı imkanlar artacaq.

İxtisaslaşmış DDoS mühafizə xidmətləri DDoS hücumlarını şəbəkənizə çatmazdan əvvəl aşkarlaya və azalda bilər. Bu xidmətlər çox vaxt trafiki öz şəbəkələri vasitəsilə yönləndirmək yolu ilə fəaliyyət göstərir, burada serverlərinizə göndərilməzdən əvvəl təhlil edilir və təmizlənir. Bu sistemlərə CloudFlare xidmətini nümunə olaraq göstərmək olar.

Veb saytlarda adətən məlumatların böyük hissəsini statik fayllar – şəkillər, JavaScript faylları, videolar, CSS faylları və s. təşkil edir. Bu məlumatları xüsusi CDN sistemlərinə

yükləməklə, saytınızı böyük bir yükdən azad etmiş olursunuz. CDN-lər DDoS hücumlarının təsirini azaltmağa kömək etməklə, geniş server şəbəkələrində böyük həcmdə trafiki qəbul edə və yaya bilər. CDN-lər haqqında növbəti bölmədə daha ətraflı məlumat verilmişdir.

Bundan başqa yaratdığınız sistemin "dözümünü" bilmək hər zaman faydalıdır. Belə olduqda siz ən azından sisteminizin böhrən həddindən məlumatlı olacaq və nə qədər hücuma tab gətirəcəyini biləcəksiniz. Bu bilgilər olduqda fərqli preventiv taktikalar düşünmək olar, xüsusi alarm sistemləri qoşmaqla yüklənmələr barədə əvvəlcədən məlumatlanmaq mümkün olar. Sistemin nə qədər yükə tab gətirəcəyini ölçmək üçün Load testləşdirmə lazımdır. Bu barədə növbəti bölmədə məlumat verilmişdir.

CDN sistemlər

Məzmun Çatdırılma Şəbəkələri (CDNs) müasir internetdə mühüm əhəmiyyət kəsb edir və məzmunu bütün dünyada istifadəçilərə tez və səmərəli şəkildə çatdırmaq üçün xüsusi rol oynayır. CDN, server və istifadəçi arasında fiziki məsafəni azaltmaqla veb səhifə məzmununun yüklənməsində gecikmələri minimuma endirməyə yönəlmiş məlumat mərkəzləri ilə birlikdə coğrafi olaraq paylanmış serverlər şəbəkəsidir. CDN istifadəçiyə məzmunu ona ən yaxın serverdən çatdırılmasını təmin edir.

CDN-lərin işləməsi veb məzmunun müxtəlif yerlərdə çoxlu surətlərinin təkrarlanmasına əsaslanır. İstifadəçi məzmuna baxmaq istədikdə CDN sistemi avtomatik olaraq istifadəçi ən yaxın serverdəki surətini göstərir. Belə surət olmadıqda məlumatı əsas serverdən əldə edib yaxındakı serverdə saxlayır.

Amazon, Microsoft, CloudFlare, Akamai CDN sistemləri təklif edən böyük şirkətlərdəndirlər. CloudFront Amazonun bulud xidməti olan AWS-in geniş istifadə edilən xidmətlərindəndir.

Tətbiqlərin load test olunması

Təhlükəsizlik kontekstində yüklənmə testi (load test), performansın azalması və ya uğursuzluq olmadan yüksək trafik həcmlərini idarə edə bilməsini təmin etmək üçün sistemin, tətbiqin və ya şəbəkənin maksimum əməliyyat qabiliyyətini müəyyən etməyə kömək edən kritik bir prosesdir.

Bu test, DDoS hücumlarına və ya sistemin təbii yolla yüklənməsinə qarşı sistemləri hazırlamaq və qorumaq üçün xüsusilə vacibdir. Sistemin imkanlarını başa düşməklə, təşkilatlar bu cür hücumlara qarşı davamlılığını gücləndirmək üçün tədbirlər həyata keçirə bilərlər.

Load test, sistemin stress altında necə fəaliyyət göstərdiyini müşahidə etmək üçün müxtəlif intensivlik səviyyələrində real istifadəçi davranışını və trafik nümunələrini simulyasiya etməklə həyata keçirilir. Əsas məqsəd DDoS hücumu zamanı istifadə oluna bilə-

cək “tıxacları”, resurs məhdudiyyətlərini və potensial uğursuzluq nöqtələrini müəyyən etməkdir.

Load test-i yerinə yetirmək üçün müxtəlif alətlər mövcuddur. Bunlara JMeter, LoadRunner, Gatling və s. göstərmək olar.

10 | Mühazirə 10

Fişinq nədir?

Fişinq – bir şirkəti/sistemi təqlid edərək saxta məlumatın göndərilməsi və istifadəçi məlumatlarının əldə edilməsidir. Məsələn, bədniyyətli şəxslərin şəxsi məlumatları, bank detalları və ya həssas məlumatları əldə etmək üçün istifadə etdikləri kiber hücum növüdür. Fişinq bir sistemdə təhlükəsizlik boşluğu taparaq onu sındırmaqdan daha asandır.

Fişinq hücumları bu gün ən çox yayılmış kibertəhlükələrdən biridir. Fişinq zamanı giriş məlumatları, kredit kartı nömrələri və ya digər fərdi məlumatlar kimi həssas məlumatları əldə etmək üçün insanları aldatmaq kimi fırıldaq üsullardan istifadə edirlər. Əsasən, bu hücumlar istifadəçilərin rəqəmsal kommunikasiyalara olan etibarından istifadə edərək, özlərini qanuni görünən e-poçtlar, sosial media mesajları, mətnlər və ya telefon zəngləri vasitəsilə təqdim edirlər.

Fişinq prosesi, mesajı alanı "hərəkətə keçməyə" sövq etmək üçün hazırlanmış saxta mesajları hədəfə göndərməklə başlayır. Bu zaman həmin istifadəçidən hansısa səbəbə görə istifadəçinin şəxsi məlumatlarını daxil etməsi tələb olunur. Mesajların mövzuları müxtəlif ola bilir. Məsələn, aldadıcı mesaj istifadəçinin təhlükəsizliyinin risk altında olduğunu və ya hesab məlumatlarını dərhal yoxlamalı olduqlarını iddia edərək, təcili şəkildə istifadəçidə hansısa linkə keçid etməsini və ya hansısa məlumatları daxil etməsini tələb edir.

Aşağıdakı şəkillərdə bu cür mesajlardan nümunələri görə bilərsiniz:

Account Information <account@google.com> show details 04:57 (4 hours ago) Filter Assistant Reply

We recently have determined that different computers have logged onto your Online Banking account, and multiple password failures were present before the logons. We now need you to re-confirm your account information to us.

If this is not completed by September 28, 2007, we will be forced to suspend your account indefinitely, as it may have been used for fraudulent purposes. We thank you for your cooperation in this manner.

To confirm your Online Banking records click on the following link:
http://0313.027447202/1/online/sslencrypt218bit/online_banking/

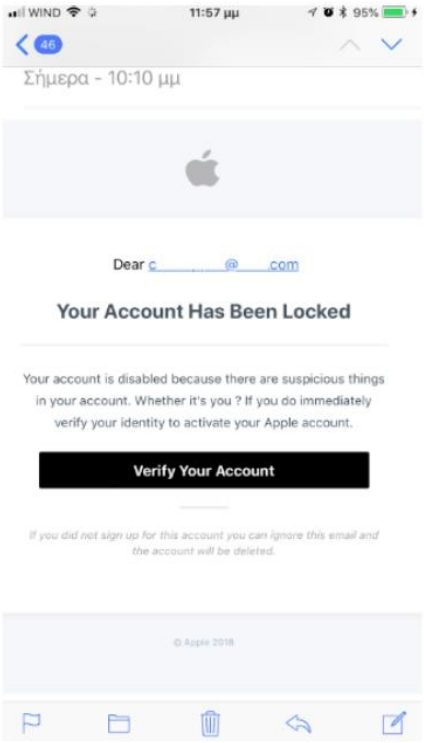
Thank you for your patience in this matter.

Bank of America Customer Service

Please do not reply to this e-mail as this is only a notification. Mail sent to this address cannot be answered.

© 2007 Bank of America Corporation. All rights reserved.

Şəkil 10.1: Özünü Bank of America kimi təqdim etməklə fişing hücumuna nümunə.



Şəkil 10.2: Apple istifadəçilərinə hədəflənmiş fişing hücumu üçün e-məktub.

Şəkil 10.3: Netflix istifadəçilərinə hədəflənmiş fişing hücumu üçün e-məktub.

Nəticə etibarı ilə, fişinq insan psixologiyasından və rəqəmsal kommunikasiyaya olan inamdan istifadə edir. Bu hücumlardan qorunmaq üçün sayıqlıq, məlumatlılıq və təhlükəsizlik qaydalarına əməl etmək lazımdır.

İndi fişinq hücumlarının fərqli növləri ilə tanış olaq

Email fişinq

Bu növ email mesajları təcili sorğulardan ibarət ola bilir. "Təcavüzkarlar" banklar, sosial media plaftormları, dövlət qurumları kimi qanuni və əsaslı mənbələrdən olduqlarını iddia edərək mesaj göndərirlər. Bu mesajları istifadəçiləri zərərli linklərə və ya şəxsi məlumatları paylaşmağa təşviq edir.

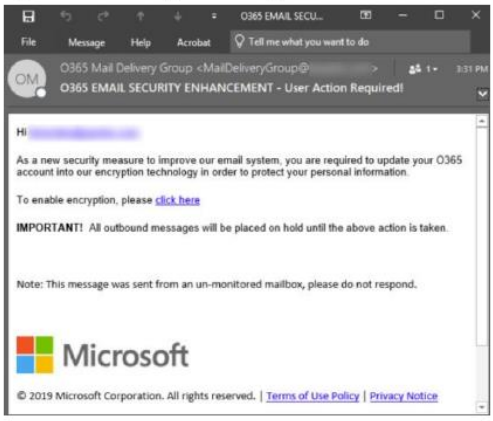
Spear fişing

Spear fişinqi geniş bir qrupu deyil konkret şəxsləri hədəf alır. Beləliklə, dələduzlar ünsiyyətlərini fərdiləşdirə və daha orijinal göstərə bilərlər. SANS İnstitutunun məlumatına görə, müəssisə şəbəkələrinə edilən bütün hücumların 95 faizi uğurlu spear fişinqin nəticəsidir.

Microsoft 365 phishing

Microsoft 365 e-poçt hesabına daxil olmaq üçün istifadə edilən bu fişinq üsulu kifayət qədər sadədir və ən çox yayılmış üsullardandır. Bu fişinq növü adətən Microsoft-dan gələn saxta e-poçt şəklində olur. E-poçtda istifadəçinin parolunu sıfırlaması lazım olduğunu izah edən məktub mətni olur və orada zərərverici link mövcud olur.

Aşağıda belə bir mesajın nümunəsi göstərilmişdir:



Şəkil 10.4: Microsoft 365 phishing

Whaling (Balina ovu)

Dələduzlar şirkət rəhbərləri kimi "böyük balıq" arxasına düşürlər. Ona görə də bu hücuma "balina ovu" deyilir. Belə hücumlar üçün dələduzlar fərqli mənbələrdən kifayət qədər ciddi məlumatlar əldə edir və böyük bir hücum planı qururlar. Şirkət rəhbərinə hücum

edib ondan məlumatları oğurlamaqla hackerlər həmin şirkətə kifayət qədər böyük ziyan vura bilirlər.

Sosial media fişinqi

Dələduzlar təfərrüatlı məlumat toplamaq üçün tez-tez sosial şəbəkələrdə və digər saytlarda qurbanlarını araşdırır və sonra hücumlarını buna uyğun şəkildə planlaşdırırlar.

Vishing (Voice Phishing) - Səsli fişinq

Səsli fişinq və ya "vishing" sosial mühəndisliyin bir formasıdır. Bu, giriş etimadnamələri kimi həssas məlumatları əldə etmək üçün nəzərdə tutulmuş saxta telefon zəngidir. Məsələn, dələduz özünü bankın texniki dəstək xidməti və ya şirkətinizin nümayəndəsi kimi göstərərək zəng edə bilər. Yeni işçilər tez-tez bu cür fırıldaqlara qarşı həssas olurlar, lakin onlar hər kəsin başına gələ bilər və getdikcə daha çox yayılır.

Smishing

Email fişinq metoduna oxşasa da, bu method sms vasitəsilə yerinə yetirilir. İstifadəçilərə həssas məlumat tələb edən mesajlar göndərilir. Bu hücum metodu WhatsApp, Viber və ya Snapchat kimi qeyri-SMS mesajlaşma proqramları vasitəsilə də həyata keçirilə bilər.

Əgər şəxs gələn mesajdan şübhələnibsə, mesaja cavab verməkdən, hər hansı linkə kliklləməkdən çəkinməli, bunun yerinə mesaji və linki yoxlamaq üçün etibarlı mənbələrədən istifadə etməlidir.

2016-cı ildə Snapchat əməkdaşı fişing hücumunun qurbanı olmuş və şirkətin vacib məlumatlarını bilmədən sızdırmışdı. Özünü şirkətin icraçı direktoru kimi təqlid edərək dələduzlar həmin əməkdaşı inandıra bilmişdi. Nəticə etibarı ilə şirkət işçilərinin əmək-haqqı məlumatları sızmışdı (https://money.cnn.com/2016/02/29/technology/snapchat-phishing-scam/index.html).

Fişinqdən müdafiə

Fişinqdən müdafiə üsulları

Yəqin ki, fərqində olmadan belə fişinq hücumları ilə qarşılaşmısınız. Bəs fişinq hücumundan necə qorunmaq lazımdır, nələri etməliyik ki, belə situasiyaların qarşısını necə almaq olar?

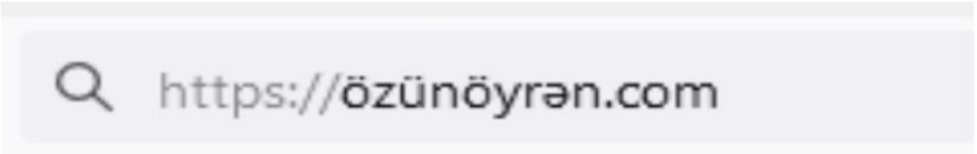
Fişinq hücumlarının birmənalı olaraq qarşısını almaq mümkün deyil. Lakin həmin hücumların fəsadlarını minimuma endirmək və fişinq qurbanı olmamaq üçün aşağıdakı tövsiyələrə əməl etmək lazımdır:

İnformasiya təhlükəsizliyi üzrə məlumatlılıq! Tanımadığınız adamlardan gələn məktubları açmayın. Əgər e-poçt ilə kimsə sizdən şifrə soruşursa heç vaxt o məktubu cavablamayın. Brauzerinizi vaxtında yeniləyin. Multi Factor Authentication üsullarını istifadə edin

IDN

IDN nədir?

Internationalized Domain Names (IDN) domen adlarında ənənəvi ASCII simvollarından əlavə müxtəlif əlifbaların simvollarından istifadə etməyə imkan verir. Məsələn, bu nümunədə özünöyrən.com ünvanında ö, ü və ə hərfləri istifadə edilmişdi.



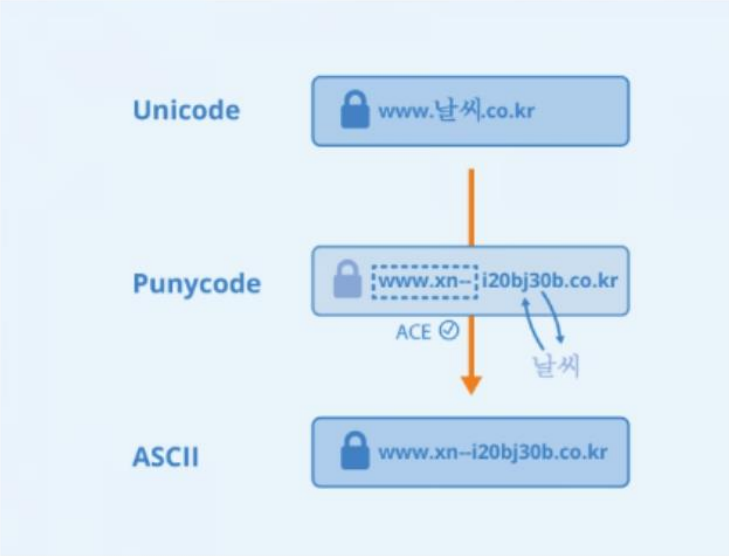
IDN çox maraqlı ideya olsa da təhlükəsizlik məsələlərində özü ilə bəzi təhlükələr gətirir, xüsusilə fişinq hücumları üçün imkanlar açır. Bu hücumlar müxtəlif əlifbalar arasındakı vizual oxşarlıqlardan istifadə edir. Məsələn, kiril əlifbasından olan p (rus əlifbasının r hərfi) latın qrafikasındakı p hərfindən vizual olaraq fərqlənmir. Bu cür nümunələr çoxdur. Təcavüzkarlar qanuni domen adlarına vizual olaraq oxşayan domen adlarını qeydiyyatdan keçirir və fişing hücumlarında istifadə edirlər.

Bu hücumlarla mübarizə aparmaq üçün müxtəlif strategiyalar həyata keçirilib. Vebbrauzerlər beynəlxalq domenlərin punycode versiyasını göstərmək kimi tədbirləri görürlər. Bu da istifadəçilərə domenin göründüyü kimi olmaya biləcəyini göstərir.

Əslində, IDN-lər internetin demokratikləşməsində mühüm rol oynasa da, həm də kibertəhlükəsizliklə bağlı yeni problemlər yaradır.

Punycode

Punycode, domen adlarında Unicode simvollarını ASCII tərəfindən dəstəklənən məhdud simvol dəstinə çevirmək üçün istifadə edilən xüsusi kodlaşdırma növüdür. İDNlərdən danışanda dedik ki, Unicode simvolaları domen adlarında istifadə oluna bilir. Amma əslində DNS sistemlərində bu adlar Punnycode şəklində yerləşdirilir, lakin brauzerlər onu bizə aydın olan şəkildə göstərirlər.



Punycode ASCİİ-yə daxil olmayan simvolları "xn–" prefiksi ilə başlayan ASCII simvolları ardıcıllığına kodlaşdırmaqla işləyir. Məsələn, Unicode domeni "münchen.com" Punycode-da "xn–mnchen-3ya.com" kimi kodlaşdırılacaq.

İDNlərlə dediyimiz problemləri bəzi hallarda brauzerlər Punnycode vasitəsilə həll edə bilirlər.

Məsələn, burada apple.com heç də ingilis dilində yazılmayıb. Burada pp hərfləri kiril qrafikasındadır. İki fərqli kodlaşdırma istifadə edildikdə brauzerlər onu şübhəli kimi qəbul edir və onu punnycode şəklində göstərərək istifadəçiləri məlumatlandırırlar.

11 | Mühazirə 11

SQL

Ümumi məlumat

SQL kimi tanınan Strukturlaşdırılmış Sorğu Dili, relyasiyalı verilənlər bazalarını idarə etmək və manipulyasiya etmək üçün standartlaşdırılmış sorğu dilidir. 1970-ci illərdə IBM şirkəti tərəfindən hazırlanmış SQL, verilənlər bazalarının idarəedilməsi üçün universal dilə çevrilmişdir. SQL istifadəçilərə məlumatları yaratmağa, dəyişdirməyə, idarə etməyə və sorğu verməyə imkan verir.

SQL-in əsasını verilənlər bazasında saxlanılan məlumatlar üzərində geniş əməliyyatlar yerinə yetirmək qabiliyyəti təşkil edir. O, istifadəçilərə sorğular vasitəsilə yeni məlumatlar daxil etməyə, mövcud məlumatları yeniləməyə, arzuolunmaz məlumatları silməyə və verilənlər bazasından məlumatları əldə etməyə imkan verir. SQL yalnız məlumatların manipulyasiyası ilə məhdudlaşmır, həm də verilənlər bazası cədvəllərinin strukturunu və onlar arasında əlaqələri müəyyən etməyə və dəyişdirməyə imkan verir.

SQL-in standartlaşdırılması dilin əsas hissəsinin MySQL, PostgreSQL, Oracle və SQL Server kimi müxtəlif verilənlər bazası sistemlərində eyni olmasını təmin etmişdir. Eyni sadə sorğunu bu bazaların hər birində yerinə yetirmək mümkündür. Ümumi sintaksis eyni olsa da funksionallığı artırmaq üçün bu sistemlərin bir-birindən fərqli əlavə imkanları da vardır.

Mühazirədəki materialı ciddi şəkildə qavramaq üçün seminar dərslərində istənilən SQL tipli verilənlər bazası ilə praktiki tapşırıqlar etmək və növbəti bölmədəki mövzular üzrə praktiki təcrübələr aparmaq vacibdir. Geniş yayılmış verilənlər bazaları sistemlərinə MySQL və PostgreSQL misal ola bilər. Bu bazalar həm pulsuzdur ,həm bir çox layihələrdə geniş istifadə olunur.

Veb təhlükəsizlikdə SQL biliklərinin rolu

Veb təhlükəsizliyi mütəxəssisləri və veb proqramçılar üçün SQL-i (Strukturlaşdırılmış Sorğu Dili) bilmək olduqca vacibdir. Son dövrlərdə NoSQL tipli bazalar xeyli istifadə olunsalar da, bir çox sistemlər, xüsusilə biznes əhəmiyyətli və maliyyə tranzaksiyaları yerinə yetirən sistemlər SQL tipli verilənlər bazaları ilə çalışır. Verilənlər bazaları istifadəçi məlumatlarından tutmuş proqram təminatının parametrləri və konfiqurasiya parametrlərinə, maliyyə məlumatlarına, analitika üçün məlumatlara qədər hər şeyi saxlayan əksər veb proqramların mərkəzindədir. SQL dili veb proqramlar və onun məlumatları arasında körpü rolunu oynayır.

Təhlükəsizlik nöqteyi-nəzərindən, SQL biliyi tətbiqləri ən çox yayılmış təhlükəsizlik zəifliklərindən biri olan SQL inyeksiya hücumlarından qorumaq üçün vacibdir. SQL sorğula-

rının necə qurulduğunu və icra edildiyini başa düşməklə, təhlükəsizlik mütəxəssisləri və tərtibatçıları təcavüzkarların zəifliklərdən necə istifadə edə biləcəyini daha yaxşı bilə və buna görə də daha effektiv müdafiə vasitələri həyata keçirə bilərlər.

Növbəti bölmədə məşhur SQL əmrləri ilə tanış olacağıq.

SQL ilə bəzi əmrlər və sorğular

SQL sorğusu verilənlər bazasından məlumat əldə etmək məqsədilə bazaya edilən sorğudur. Bu sorğular SQL-in sintaksis qaydalarına əməl edir və istifadəçilərə verilənlər bazasında saxlanılan məlumatlar üzərində geniş əməliyyatlar həyata keçirməyə imkan verir. Əməliyyatlar verilənlərin əldə edilməsinə (SELECT), mövcud məlumatların dəyişdirilməsinə (UPDATE və DELETE sorğuları), verilənlər bazasının özünün strukturunun dəyişdirilməsinə (CREATE, ALTER və DROP əmrləri) aid ola bilərlər.

SQL sorğuları sorğunu emal edən və tələb olunan məlumatı istifadəçiyə qaytaran və ya tələb olunan hərəkəti yerinə yetirən verilənlər bazası idarəetmə sisteminə (DBMS) qarşı yerinə yetirilir. Yəni sorğu yazılır və həmin sistemə xüsusi drayver vasitəsilə ötürülür. Məsələn, siz Java dilində yazılmış kod ilə MySQL bazasına sorğu göndərərkən, MySQL-in Java üçün drayverini istifadə etməlisiniz. Yazılan sorğu drayver vasitəsilə MySQL sisteminə ötürülür. Dəqiq SQL sorğuları hazırlamaq bacarığı həm funksional, verilənlərə əsaslanan veb proqramların yaradılması, həm də onların zərərli SQL kodu inyeksiyasına qarşı təhlükəsizliyini təmin etmək üçün vacib bilikdir. SQL inyeksiyası təhlükəsizlik baxımından yeganə problem deyil. Həmçinin sorğuların optimal işləməsi əlçatanlığın təmin olunması üçün vacibdir.

İndi isə gəlin bəzi SQL əmrlərini araşdıraq. Təsəvvür edin ki, veb saytda belə bir məlumatı əks etdirmək lazımdır:

Adı	Soyadı	Təvəllüd	Qiymət 1	Qiymət 2	Qiymət 3	Qiymət 4
Orxan	Əliyev	1992	100	78	89	91
Leyla	Əhmədli	1993	82	95	87	100
Məmməd	Rəhimzadə	1994	94	99	93	100
Nərmin	Məmmədova	1992	89	100	87	74
Aynur	Vəlizadə	1991	76	100	86	98

Burada tələbələrin adları, soyadlar, təvəllüd və qiymətləri yer alıb. İlk olaraq belə bir cədvəli MySQL ilə yaratmağı öyrənək.

CREATE TABLE əmri

SQL-də CREATE TABLE əmri verilənlər bazasında yeni cədvəl yaratmaq üçün istifadə olunur. Cədvəl sütun və sətirlərdən ibarətdir. Cədvəllər relyasiyalı verilənlər bazalarında verilənlərin saxlandığı əsas strukturdur. CREATE TABLE əmri hər bir sütunun adını və onların tiplərini, cədvəlin sxemini, habelə həmin sütunlardakı məlumatların riayət etməli olduğu hər hansı məhdudiyyətləri (qaydaları) müəyyən etməyə imkan verir.

Nümunə üçün aşağıdakı CREATE TABLE əmrinə baxaq:

```
CREATE TABLE IF NOT EXISTS `telebeler` (
  `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
  `ad` varchar(200) NOT NULL,
  `soyad` varchar(200) NOT NULL,
  `tevellud` int(4) unsigned,
  `qiymet1` int(3) unsigned,
  `qiymet2` int(3) unsigned,
  `qiymet3` int(3) unsigned,
  `qiymet4` int(3) unsigned,
  PRIMARY KEY (`id`)
) DEFAULT CHARSET=utf8;
```

Bu SQL ifadəsi verilənlər bazasında "tələbələr" adlı yeni cədvəl yaradır. O, sütunları, onların məlumat növlərini və məhdudiyyətləri göstərərək cədvəlin strukturunu müəyyən edir.

Birinci sətirdəki SQL əmri verilənlər bazasında "tələbələr" adlı cədvəl olmadıqda onu yaratmağa cəhd etməsini təmin edir. Burada "tələbələr" yaradılacaq cədvəlin adıdır. İkinci sətirdən başlayaraq sütunların adları və tipləri göstərilmişdir.

İNSERT əmri

SQL-də INSERT əmri cədvələ bir və ya daha çox yeni məlumat sətirləri əlavə etmək üçün istifadə olunur. O, verilənlər bazalarını ilkin yaradıldıqdan sonra yeni məlumatlarla doldurmağa imkan verən verilənlərin manipulyasiya dili (DML) əməliyyatlarının mühüm hissəsidir.

Əvvəlki bölmədə tələbələr haqqında məlumatların, o cümlədən onların adları, doğum ili və qiymətləri saxlamaq üçün nəzərdə tutulmuş "tələbələr" adlı cədvə yaratdıq. Bu cədvələ yeni məlumat daxil etmək üçün aşağıdakı INSERT əmrindən istifadə edə bilərsiniz:

```sql
INSERT INTO `telebeler` (`ad`, `soyad`, `tevellud`, `qiymet1`, `qiymet2`, `qiymet3`, `qiymet4`)
VALUES ('Orxan', 'əliyev', 1992, 100, 78, 89, 91);
```

Burada əvvəlcə məlumat daxil ediləcək sütunların adları, daha sonra eyni ardıcıllıqla VALUES bölməsinin daxilində onlara verilən qiymətlər yazılır. Gördüyünüz kimi, "id" sütununu burada yazmamışıq. CREATE TABLE əmrinə diqqət etsəniz orada "id" sütunu üçün {AUTO_İNCREMENT} seçimi əlavə edilibdir. Bu o deməkdir ki, cədvələ yeni məlumat daxil edildikdə həmin sütuna sistem özü bir qiymət verəcək və növbəti dəfə həmin qiymətdən bir vahid çox qiymət verəcək. Beləliklə, onun qiymətləri 1,2,3,... kimi davam edəcək. Bu bir növ sətirlərin sıra nömrəsi və ya daha dəqiq desək hər bir sətirin unikal identifikasiya nömrəsi olacaqdır.

İNSERT əmri ilə bir deyil daha çox məlumatı bir dəfəyə yaratmaq da olar.

```sql
INSERT INTO `telebeler` (`ad`, `soyad`, `tevellud`, `qiymet1`, `qiymet2`, `qiymet3`, `qiymet4`)
VALUES
  ('Orxan', 'əliyev', 1992, 100, 78, 89, 91),
  ('Leyla', 'əhmədli', 1993, 82, 95, 87, 100),
  ('Məmməd', 'Rəhimzadə', 1994, 94, 99, 93, 100),
  ('Nərmin', 'Məmmədova', 1992, 89, 100, 87, 74),
  ('Aynur', 'vəlizadə', 1991, 76, 100, 86, 98);
```

Bu nümunədə bir komanda ilə 5 tələbə haqqında məlumat daxil edilir.

SELECT əmri

SELECT əmri mövcud cədvəldəki məlumatları əldə etmək üçündür. Aşağıdakı nümunəyə baxaq:

```sql
SELECT * FROM telebeler;
```

id	ad	soyad	tevellud	qiymet1	qiymet2	qiymet3	qiymet4
1	Orxan	əliyev	1992	100	78	89	91
2	Leyla	əhmədli	1993	82	95	87	100
3	Məmməd	Rəhimzadə	1994	94	99	93	100
4	Nərmin	Məmmədova	1992	89	100	87	74
5	Aynur	vəlizadə	1991	76	100	86	98

SELECT əmri ilə bir deyil bir neçə cədvəldən də məlumat əldə etmək mümkündür. Növbəti mühazirədə SELECT əmri barədə bir qədər daha detallı öyrənəcək və orada təhlükəsizlik nöqteyi nəzərdən diqqət etməli olduğumuz məqamları araşdıracağıq.

UPDATE əmri

UPDATE əmri cədvəldəki mövcud məlumatları dəyişdirmək üçün istifadə olunur. Bu əmr, WHERE bəndi ilə müəyyən edilmiş müəyyən meyarlara cavab verən bütün sətirlər üçün bir və ya bir neçə sütunun dəyərini dəyişməyə imkan verir.

Aşağıdakı nümunədə id nömrəsi 5 olan tələbənin soyadı dəyişdiriləcək Vəlizadə (böyük hərflə) olaraq yazılır.

```
UPDATE telebeler SET soyad='Vəlizadə' WHERE id=5;

SELECT * FROM telebeler;

| id |     ad |     soyad | tevellud | qiymet1 | qiymet2 | qiymet3 | qiymet4 |
|----|--------|-----------|----------|---------|---------|---------|---------|
|  1 |  Orxan |    Əliyev |     1992 |     100 |      78 |      89 |      91 |
|  2 |  Leyla |   Əhmədli |     1993 |      82 |      95 |      87 |     100 |
|  3 | Məmməd | Rəhimzadə |     1994 |      94 |      99 |      93 |     100 |
|  4 | Nərmin | Məmmədova |     1992 |      89 |     100 |      87 |      74 |
|  5 |  Aynur |  Vəlizadə |     1991 |      76 |     100 |      86 |      98 |
```

DELETE əmri

SQL-də DELETE ifadəsi WHERE bəndi ilə müəyyən edilmiş şərt əsasında cədvəldən bir və ya bir neçə sətir silmək üçün istifadə olunur. Məsələn aşağıdakı nümunədə id nömrəsi 5 olan element silinmişdir.

```
DELETE FROM telebeler WHERE id=5;

SELECT * FROM telebeler;

| id |     ad |     soyad | tevellud | qiymet1 | qiymet2 | qiymet3 | qiymet4 |
|----|--------|-----------|----------|---------|---------|---------|---------|
|  1 |  Orxan |    Əliyev |     1992 |     100 |      78 |      89 |      91 |
|  2 |  Leyla |   Əhmədli |     1993 |      82 |      95 |      87 |     100 |
|  3 | Məmməd | Rəhimzadə |     1994 |      94 |      99 |      93 |     100 |
|  4 | Nərmin | Məmmədova |     1992 |      89 |     100 |      87 |      74 |
```

DROP əmri

SQL-də DROP TABLE ifadəsi cədvəli və onun tərkibində olan bütün məlumatları verilənlər bazasından birdəfəlik silmək üçün istifadə olunur. Bu əmr təkcə cədvəlin içindəki məlumatları deyil, həm də cədvəl strukturunun özünü silir. Cədvəl atıldıqdan sonra ehtiyat nüsxəsi olmadıqda onu bərpa etmək mümkün deyil. DROP TABLE əmri ehtiyatla istifadə edilməlidir.

12 | Mühazirə 12

SQL injection nədir?

Bu mühazirədə biz veb təhlükəsizlikdə davamlı şəkildə rast gəlinən və kritik zəifliklərdən biri SQL Injection haqqında danışacağıq. Müştəridən serverə daxil olan giriş (input) vasitəsilə SQL sorğularını manipulyasiya edən bu tip hücumlar verilənlər bazası ilə qarşılıqlı əlaqədə olan istənilən proqram üçün əhəmiyyətli təhlükə yaradır. SQL inyeksiyasının mahiyyəti zərərli SQL kodunu sorğuya daxil etməklə və ya "inyeksiya etməklə" proqram təminatına zərər vurmaqdan ibarətdir.

SQL injection hücumunun mərkəzində təcavüzkarın verilənlər bazasına göndərilən SQL əmrlərinə dəyişiklik etmək imkanları dayanır. Bu, verilənlər bazasında saxlanılan məlumatlara icazəsiz girişə səbəb ola bilir. Bundan başqa vəziyyətdən asılı olaraq SQL inyeksiyası hücumu ilə verilənlər bazasını korlamaq mümkün olur. SQL ilə bəzi VBİSlərdə sistem əmrlərini yerinə yetirmək də mümkündür. Belə olduqda SQL inyeksiyası hücumu ilə bir-başa sistem əmrlərini yerinə yetirməklə sistemi bütövlükdə ələ keçirmək mümkündür.

Bu mühazirədə biz SQL inyeksiyası haqqında məlumat əldə edəcək, onun yerinə yetirilməsi yollarını araşdıracaq və SQL inyeksiyası hücumlarının qarısının alınması yolları barədə söhbət açacağıq.

SQL injection ilə nümunə

Aşağıdakı koda nəzər yetirək. Bu kod fraqmenti NodeJS-in express kitabxanasını istifadə edərək veb serverin necə qurulmasını və istifadəçinin autentifikasiyası üçün MySQL verilənlər bazası ilə əlaqə yaratmağı təsvir edir. İlk üç sətir uyğun olaraq express, mysql və router kitabxanalarını module əlavə edir ki, onlardan yararlanmaq mümkün olsun.

```
1  var express = require('express');
2  var mysql = require('mysql');
3  var router = express.Router();
4
5  router.post('/', function (req, res) {
6    var con = mysql.createConnection({host: "localhost", user: "adil", password: "passw0rd!", database: "m12"});
7
8    con.connect(function (err) {
9      const { username, password } = req.body;
10     con.query(`SELECT * FROM users WHERE username='${username}' AND password='${password}'`,
11               function (err, result, fields) {
12                 if (result.length>0) {
13                   res.send("<h1>Xoş gəldiniz!</h1>");
14                 } else {
15                   res.send("<h1>İstifadəçi adı və ya parol səhvdir.</h1>");
16                 }
17               });
18   });
19 });
20
21 module.exports = router;
```

Beşinci sətirdə POST metodu ilə sorğuları qəbul etmək üçün APİ elan edilir. POST sorğu-

su qəbul edildikdə, burada müəyyən edilmiş funksiya yerinə yetiriləcək. "req" sorğu obyektini, "res" isə onun əsasında formalaşan cavab obyektini təmsil edir. Növbəti sətirdə MySQL verilənlər bazası ilə əlaqə yaradılır və bu əlaqəni özündə saxlayan bağlantı "con" dəyişəninə mənimsədilir. Sonrakı sətirdə isə "con" dəyişənini istifadə etməklə MySQL ilə real bağlantı yaradılır. Bağlantı yaradılarkən "req" sorğu obyektindən gələn "body" dəyişəni açılaraq "username" və "password" dəyişənlərinə uyğun qiymətləri mənimsədir.

Onuncu sətirdə əsas diqqət etməli məqam başlayır. Burada "con" bağlantısı MySQL-ə SELECT sorğusu göndərir. Bu sorğuya əsasən "users" cədvəlindən istifadəçi haqqında məlumat əldə edilir. Konkret hansı istifadəçini bildirmək üçün sorğunun WHERE bəndində müvafiq kriteriya əlavə edilmişdir. Burada "req" parametrindən gəlmiş "username" və "password" dəyişənlərinin qiymətləri sorğuya əlavə edilir. Nəticə etibarı ilə bu sorğu istifadəçi adı və şifrəyə görə istifadəçini bazadan tapmaqdan ibarətdir. Sonrakı sətirdə əgər əldə edilən nəticələrin sayı 0-dan çoxdursa, başqa sözlə desək əgər nəticə varsa "Xoş gəlmisiniz" mətni cavab olaraq göndərilir. Yox, əgər 0-a bərabərdirsə, bu o deməkdir ki, verilmiş istifadəçi adı və şifrə ilə bazada heç bir istifadəçi tapılmadı. Beləliklə bu halda da müvafiq mesaj cavab olaraq göndərilir.

Gəlin bu koddakı SQL sorğusuna daha dərindən diqqət edək. Təsəvvür edin ki, istifadəçi adı "orxan" onun şifrəsi isə "o1234@" kimi daxil edilib. O zaman sorğu bu şəkildə olacaq:

```
1 SELECT * FROM users WHERE username='orxan' AND password='o1234@'
```

Bu sorğu yerinə yetiriləcək və verilənlər bazasında istifadəçi adı və şifrəsi "orxan" və "o1234@" olan element varsa həmin element (sətir) əldə ediləcək. Yoxdursa o zaman boş massiv qayıdacaq. Məntiqi olaraq düzgün sorğu olduğunu görürük.

Təsəvvür edin ki, hər hansı bir haker istifadəçi adı Orxan olan birinin sitemdə olduğunu bilir. Və o, veb saytda istifadəçi adı kimi "orxan" və şifrə olaraq "1' OR '1'='1" yazır. Belə olduqda yekun sorğu aşağıdakı şəkili almış olur:

```
1 SELECT * FROM users WHERE username='orxan' AND password='1' OR '1'='1'
```

Bu sorğu istifadəçi adı "orxan" şifrəsi isə "1" olan istifadəçini tapmaq istəyir, lakin şərtlərə həm də OR vasitəsilə (məntiqi vəya) '1'='1' ifadəsini də birləşdirib. Bu bərabərlik isə hər zaman doğrudur. Hər zaman doğru olan bərabərliyi OR ilə ifadəyə birləşdirdikdə yekun ifadə hər zaman doğru olur. Beləliklə haker şifrəni bilmədən sistemə daxil ola bilər.

Başqa nümunəyə baxaq. Təsəvvür edin, hacker şifrə əvəzinə "'; DROP TABLE users; —" yazır. Bu zaman sorğu aşağıdakı kimi olacaq:

```
1 SELECT * FROM users WHERE username='orxan' AND password=''; DROP TABLE users; --'
```

Burada şifrəni göstərən bənddən sonra ";" işarəsi qoyularaq sorğunun bitdiyi işarə edilir. Ardınca yeni sorğu başlayır "DROP TABLE users" və ";" işarəsi ilə bu sorğu da dərhal bitmiş göstərilir. Sonra isə iki ədəd defis işarəsi qoyulub. MYSQL sintaksisində qoşa defis işarəsi şərh bildirir. Yəni kodun ondan sonrakı hissəsi çalışmayacaq.

Bu sorğu icra olunanda əgər tətbiqi işə salan sistem istifadəçisinin səlahiyyəti varsa cədvəl silinəcək. Beləliklə proqram təminatı bir balaca sorğu ilə korlanacaq. Bu əlçatanlığın bir-başa korlanması deməkdir.

Yuxarıdakı nümunələrdə gördüyünüz kimi, hacker inputlarda cüzi dəyişiklik edərək öz kodunu sistemə yeritmiş oldu. Bu cür hücum növünə SQL İnjection deyilir.

SQL injection hücumunun qarşısının alınması

Əvvəlki bölmələrdə SQL injection hücumunun nə olduğunu və necə çalışdığını öyrəndiniz. İndi isə bu hücumların qarşsının alınması yollarını araşdıraq.

SQL inyeksiyasının qarşısını almaq üçün elə etmək lazımdır ki, istifadəçidən gələn heç bir input sorğunun dəyişməsinə səbəb olmasın. Bunu etməyin fərqli yolları var. Məsələn, yuxarıdakı problem ondan ibarət idi ki, " ' " işarələrinin köməyi ilə sorğunu dəyişmək mümkün olur. Müxtəlif fərqli simvollar da sorğunu dəyişməyə kömək edə bilər. Əgər siz proqramınızda elə bir alqoritm qursanız ki, bu işarələri qəbul etməsin, yaxud onları silsin o zaman SQL inyeksiyası hücumunun qarşısını almaq olar.

Bu simvolları hər dəfə filtrləmək əlavə kod yazmaq tələb edir. Və digər tərəfdən proqramın üzərində işləyərkən asanlıqla insanın fikrindən çıxa bilər. Ona görə də SQL inyeksiyasına qarşı ən yaxşı üsullardan biri parametrləşdirilmiş sorğuların istifadəsidir. Bu üsul ilə SQL sorğusu dəyişmir və istifadəçidən gələn verilənlər həmin hissəyə parametr kimi daxil olur. Bu parametrin adi sətirlərin birləşməsindən fərqi odur ki, lazım olan filtrləməni özü edir. Parametrləşmiş ifadələr istifadə etdiyiniz proqramlaşdırma dili üçün lazım olan VBİS-in drayveri səviyyəsində çalışır. Proqramçılar sadəcə olaraq bütün girişləri parametr kimi etməklə SQL inyeksiyası riskini azaltmış olurlar. Əksər müasir verilənlər bazası idarəetmə sistemləri (DBMS) və proqramlaşdırma dilləri parametrləşdirilmiş sorğuları dəstəkləyir.

Gəlin, fərqli proqramlaşdırma dillərində yazılmış bir neçə nümunəyə baxaq.

NodeJS (JavaScript)

Aşağıdakı kodda 1-ci sətirdə JavaScript üçün mysql modulu çağrılır. Daha sonra ondan istifadə edərək 2-7-ci sətirlərdə verilənlər bazasına qoşulma parmetrləri verilmişdir. Tə-

bii ki, real layihələrdə bu cür bəsit istifadəçi adı və şifrələr olmur və bir-başa kodun içində yazılmır.

9-cu sətirdə bazaya qoşulma prosesi baş verir ki, bazaya sorğular göndərilə bilsin. 14 və 15-ci sətirlərdə nümunə olaraq giriş parametrləri verilmişdir. Kodda onlar statik dəyər kimi verilsə də real sistemlərdə onlar parametrlər olmalıdır. Ona görə də bu koda yalnız nümunə olaraq yanaşmaq lazımdır.

SQL inyeksiya hücumunun qarşısını almaq üçün əsas iş 16 və 18-ci sətirlərdə baş verir. 16-cı sətirdə "query" dəyişəninə SQL sorğunun mətni mənimsədilib. Gördüyünüz kimi orada username və password dəyişənlərinin qiymətləri olmalı yerdə "?" işarəsi ilə parametrin olmalı olduğu verilib. 18-ci sətirdə query mətni parametr kimi verilir və "?" sual işarələrinin yerində olmalı olan parametrlərin qiymətləri massiv kimi göstərilir. Beləliklə "mysql" modulundakı "query" funksiyası parametrləri təhlükəsiz şəkildə ötürür və SQL inyeksiyası hücumunun qarşısını alır.

```
1  const mysql = require('mysql');
2  const connection = mysql.createConnection({
3    host: 'localhost',
4    user: 'user',
5    password: 'password',
6    database: 'database'
7  });
8
9  connection.connect(err => {
10   if (err) throw err;
11   console.log('Connected!');
12 });
13
14 const username = 'sampleUser';
15 const password = 'samplePassword';
16 const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
17
18 connection.query(query, [username, password], (err, results) => {
19   if (err) throw err;
20   console.log(results);
21 });
22
23 connection.end();
24
```

Bir şeyi də nəzərə alın ki, inyeksiya hücumunun qarşısının alınması adətən təhlükəli simvolların – yəni sorğu dəyişə biləcək simvolların ləğv edilməsi əsasında çalışır. Böyük ehtimal ki, yuxarıdakı mysql modulunun iş prinsipi də buna əsaslanır. Əgər belədirsə o zaman o modulun özünün də boşluğu ola bilər. Belə olduqda bizim sistemin yenə də SQL inyeksiya hücumlarına qarşı zəifliklər olacaqdır. Bunun qarşısının alındığından əmin olmaq üçün ilk olaraq inanılmamış və sınanılmamış kitabxana və ya modulları istifadə etməyin. Həmçinin layihə istismara çıxmadan öncə mütləq nüfuzetmə testləri yerinə ye-

tirilməlidir.

Aşağıda oxşar kodun fərqli dillərdə olan versiyaları verilmişdir. Mühazirələrimizdə daha çox JavaScript dili üzrə çalışsaq da digər dillərin sintaksisi ilə də tanış olmaq və SQL inyeksiyası kimi konseptlərin həmin dillərdə də necə baş verdiyi və qarşısının necə alındığını bilmək vacibdir.

Java

Məntiq etibarı ilə Java dilində yazılmış aşağıdakı həll elə yuxarıda bəhs etdiyimiz digər həllin məntiqinə oxşayır. Sadəcə yerinə yetirilməsi bir qədər fərqli şəkildə həyata keçirilmişdir.

Məsələn, Java dilində MySQL-ə qoşulmaq üçün JDBC adlı aralıq sistemdən istifadə edilir. JDBC Java sistemləri ilə verilənlər bazası arasında universal körpü rolunu oynayır.

```java
import java.sql.*;

public class Main {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/database";
        String user = "user";
        String password = "password";

        String query = "SELECT * FROM users WHERE username = ? AND password = ?";

        try (Connection conn = DriverManager.getConnection(url, user, password);
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString(1, "sampleUser");
            stmt.setString(2, "samplePassword");

            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                // process result set
                System.out.println("User found.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Java dili üçün fərqli kitabxanaların öz sorğu mexanizmləri də vardır. Məsələn, ən çox istifadə edilən Spring freymvörkündə verilənlər bazaları ilə işləmək üçün çoxlu mexanizmlər var və onlar həm proqramçının işini asanlaşdırır həm də SQL inyeksiyası hücumlarının qarşısını avtomatik alır.

Təbii ki, bu cür freymvörklər də yuxarıda dediyimiz təhlükəsizlik boşluqlarından sığortalanmayıb. Sadəcə onlar bir çox böyük şirkətlər tərəfindən istifadə edilir və əgər belə bir

boşluq olarsa adətən dərhal bütün icmaya xəbərdarlıq edilir.

C#

C# dilində yanaşma Java dilindəki yanaşmaya bənzərdir. Oxunaqlılıq üçün aşağıdakı kodda 9-cu və 14-15-ci sətirlərdə dəyişənlərin "@" işarəsi ilə annotasiya olduğuna diqqət edək.

```
1  using System;
2  using System.Data.SqlClient;
3
4  class Program
5  {
6      static void Main()
7      {
8          string connectionString = "Server=localhost;Database=database;User Id=user;Password=password;";
9          string query = "SELECT * FROM users WHERE username = @username AND password = @password";
10
11         using (SqlConnection conn = new SqlConnection(connectionString))
12         {
13             SqlCommand cmd = new SqlCommand(query, conn);
14             cmd.Parameters.AddWithValue("@username", "sampleUser");
15             cmd.Parameters.AddWithValue("@password", "samplePassword");
16
17             conn.Open();
18
19             SqlDataReader reader = cmd.ExecuteReader();
20
21             while (reader.Read())
22             {
23                 Console.WriteLine("User found.");
24             }
25
26             reader.Close();
27         }
28     }
29 }
```

Python

Aşağıdakı kod isə Python dilində yazılmışdır. Python dilində MySQL bazasına qoşulmaq üçün pymysql modulundan istifadə edilir.

Burada sorğunun mətnində parametlər sətirlərin formatlanmasına uyğun %s işarələri ilə verilmişdir. Sorğunun icrası zamanı həmin parametrlərin dəyərləri Pythonda "tuple" tipli data strukturu kimi ötürülür.

```python
import pymysql

connection = pymysql.connect(host='localhost',
                             user='user',
                             password='password',
                             database='database')

try:
    with connection.cursor() as cursor:
        sql = "SELECT * FROM users WHERE username = %s AND password = %s"
        cursor.execute(sql, ('sampleUser', 'samplePassword'))
        result = cursor.fetchone()
        if result:
            print("User found.")
finally:
    connection.close()

```

13 | Mühazirə 13

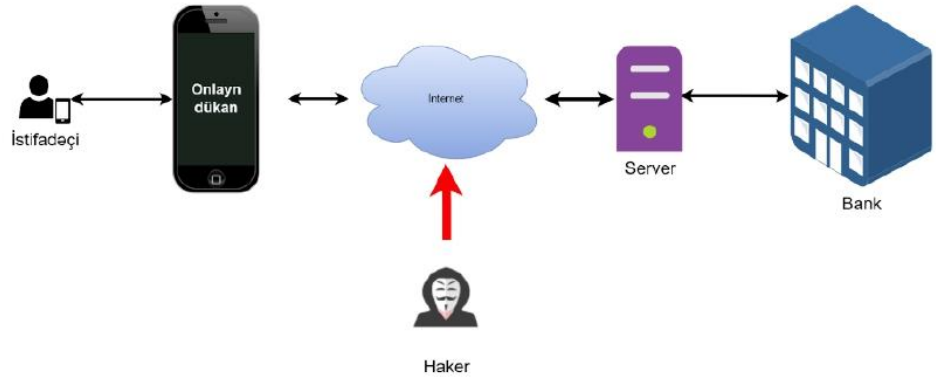
Kriptoqrafiya

Kriptoqrafiya barədə qısa məlumat

İnternet üzərindən bir məhsul aldığınız zaman, kredit kartınız haqqında məlumatları daxil edirsiniz. Bununla da kart haqqında məlumatlar satıcının veb saytındakı serverinə və ya üçüncü tərəf ödəmə xidmətinə ötürülür. İnternet açıq şəbəkədir və hər kəs oradan müəyyən məlumatları əldə edə bilər. Buna görə kredit kartı nömrəniz şifrələnmədən göndərilsə, ixtiyari şəxs üçün kartınızın nömrəsini əldə etmək asan olar. Bunu etməyin yollarını sniffrələr haqqında altıncı mühazirədə danışarkən öyrənmişdiniz.

Kart məlumatlarınız bir dələduzun əlinə keçərsə hesabınızdan qeyri-qanuni yolla istifadə edərək məhsul və xidmətlər almağa başlaya bilər.

Mühazirənin bir hissəsi müəllifin tərcümə etdiyi "Alqoritmlərin Sirri" kitabından götürülmüşdür.



Şəkil 13.1: Hakerlər açıq kanal ilə hərəkət edən məlumatları asanlıqla əldə edə bilər.

Təbii ki, kiminsə məhz sizi izləməsi ehtimalı aşağıdır. Çətin ki, kimsə hazırda sizin İnternet üzərindən kredit kartı nömrəsinə bənzər bir data göndərməyinizi gözləyir. Amma yenə də kredit kartı nömrənizi İnternet üzərindən göndərdiyiniz zaman onu şifrələmək daha etibarlı olardı. Böyük ehtimal ki, belə də edirsiniz. Təhlükəsiz veb saytlardan istifadə edirsinizsə - URL ünvanı "http:"əvəzinə "https:"ilə başlayır. Belə olduqda brauzeriniz göndərdiyi bütün məlumatları şifrələyir (https protokolu həm də "autentifikasiya"nı təmin edir ki, bağlandığınız saytın həqiqətən sizə lazım olan sayt olduğuna əmin olasınız). Şifrələmə və de-şifrələmə prosesləri birlikdə kriptoqrafiyanın təməlini təşkil edir.

Şəxsi kredit kartı nömrəmi qorumağın vacib olduğunu hesab etsəm də, qlobal miqyasda mənim kredit kart nömrəmin bir o qədər də əhəmiyyətli olmadığının fərqindəyəm. Kim-

sə mənim kredit kartı nömrəmi oğurlayırsa, dövlətimizin təhlükəsizliyi risk altına düşməz. Ancaq kimsə dövlət əhəmiyyətli təlimatların diplomatlara göndərilən zamanı əldə edə bilərsə və ya hərbi əməliyyatlarla bağlı məlumatlara girişi olarsa milli təhlükəsizliyimiz həqiqətən də risk altına düşər. Bu səbəbdən yalnız məlumatları şifrələmə və de-şifrələmə yollarının özləri deyil, həm də onların sındırılmasının olduqca çətin olmasını təmin etmək vacibdir.

Cari mühazirədə şifrələmənin və de-şifrələmənin təməlində duran bəzi əsas fikirləri nəzərdən keçirəcəyik. Kriptoqrafiya kifayət qədər geniş bir sahədir. Bu mühazirədə onun baza anlayışlarına toxunacağıq və veb təhlükəsizlikdə bizim köməyimizə çatacaq biliklərə yiyələnəcək, onun SSL kimi protokollarda rolu barədə danışacağıq. Bu mühazirədə öyrənilən kriptoqrafik üsullar ciddi kriptoqrafiya sistemi qurmağa kifayət etmir. Bunun üçün daha çox biliklərə yiyələnmək lazımdır ki, bu fənnin əhatəsinə daxil deyil. Məşhur kriptoqraf Ron Rivestin (bu fəsildə daha sonra görəcəyimiz RSA kriptosisteminin ixtiraçılarından biri) belə bir deyimi var: "Ümumiyyətlə kriptoqrafiya bir döyüş sənəti yarışmasına bənzəyir və praktikada uğurlu istifadə üçün ən müasir yanaşmaları bilməlisən". Bu mühazirədə məlumatları şifrələmək və de-şifrələmək üçün yalnız bəzi alqoritmlərlə tanış olacağıq.

İlk olaraq bəzi əsas terminlərlə tanış olaq. Kriptoqrafiyada orijinal məlumatları açıqmətn (ing. plaintext), şifrələnmiş versiyanı isə şifrmətn (ing. ciphertext) adlandırırıq. Buna görə şifrələmə mətni şifrmətn, de-şifrələmə isə şifrmətni yenidən orijinal açıqmətnə çevirməkdir. Çevirmək üçün lazım olan məlumatlar kriptoqrafik açar adlanır.

Təsvir olunan bu proseslərin hamısı brazuer proqramı tərəfindən avtomatik həyata keçirilir.

Əvəzetmə şifrləri

Sadə əvəzetmə şifrində sadəcə bir hərfin başqa bir hərflə əvəz edilməsilə mətni şifrələnirsiniz. Əvəzləməni tərsinə etməklə isə şifrələnmiş mətni deşifrələyirsiniz. Yuli Sezar öz generalları ilə "sürüşdürməklə şifrləmə"dən istifadə edərdi. O, hər bir hərfi əlifba sırasında özündən üç mövqe sonra gələn hərflə əvəz edər, əlifbanın son hərfləri üçün isə əlifbanın əvvəlindən başlamaqla sayaraq əvəzetməni yerinə yetirərdi. Məsələn, 26 hərfdən ibarət ingilis əlifbasında A-nın yerinə D, Y-nin yerinə B (Y-dən sonra Z, sonra A və B) yazılacaqdı. Sezarın "sürüşdürməli şifrəsi"ndə, bir generalın daha çox əsgərə ehtiyacı varsa, açıqmətni şifrələyib "Mənə əlavə yüz əsgər göndər" əvəzinə "Öğpğ ğoçağ bzc ğuxğt xspfğt" şifrmətini göndərirdi. Bu şifrəli mətni aldıqdan sonra Sezar, hər məktubu əlifbada üç mövqe əvvəldə gələn hərflə əvəz edərdi və orijinal açıqmətni əldə etmiş olurdu (Sezarın dövründə, əlbəttə ki, mesaj, dövrün Latın əlifbasından istifadə edərək Latın dilində olurdu).

Şəkil 13.2: Sezar şifrələməsi

Belə bir mesajı ələ keçirsəniz və onun “sürüşdürməli şifrələmə” ilə şifrələndiyini bilirsinizsə, sürüşmə miqdarını (açarını) əvvəlcədən bilməsəniz belə onu deşifrələmək olduqca asan olar. Açıqmətn bir məna kəsb edənə kimi bütün mümkün açarları sınamaqla şifri sındırmaq mümkündür. Məsələn 32 hərfi olan əlifba üçün maksimum 31 fərqli sürüşməni istifadə etmək kifayətdir.

Ola bilər ki, şifrələmədəki qanunauyğunluq heç də sürüşmələrlə deyil, ya da onun sürüşmələrlə olduğunu bilmirik. Tutaq ki, açıqmətn “Mənə əlavə yüz əsgər göndər” şifrmətninə çevrilmiş və nəticə “Üsvs suıhs xğm srysg yəvqsg” olmuşdur. Şifrmətndə “s” hərfi çox görünür və onun açıqmətnində isə ən çox rast gəlinən hərf “ə” hərfidir, bu isə Azərbaycan dilində tezliyi ən yüksək olan hərflərdən biridir. Sonra şifrəli mətndə dördhərfli “üsvs” sözünü görüb təxmin etməklə “ü” və “v” hərflərinin orijinalını çözə bilərsiniz. Bu minvalla digər hərfləri də təxmin yolu ilə çözüb açıqmətni əldə etmək olar.

Əlbətdə ki, kredit kartı nömrələrini şifrələyirsinizsə, o zaman hərflərin tezliyi barədə çox da narahat olmağa ehtiyac yoxdur. Ancaq on rəqəmdən söhbət gedirsə bir rəqəmin digərinə çevrilməsi üçün kombinasiyaların sayı 10! olur və bu da 3 628 800 edir. Kompüter üçün heç də böyük ədəd deyil, xüsusən 1016 mümkün kredit kartı nömrəsi ilə (16 rəqəmli kredit kartı) müqayisə edildikdə. Dələduzlar bu işi avtomatlaşdırmaqla şifrmətndən kredit kartının nömrəsini tapa və müəyyənləşdirə bilərlər.

Gəlin praktiki bir nümunə əsasında əvəzetmə şifrlərinin nə üçün etibarsız olduğunu araşdıraq. Təsəvvür edin ki, məlumatları hansısa bir açar ilə şifrələyib serverinizə göndərirsiniz. Açarı yalnız siz və server bilir. Ona görə də göndərilən məlumatı yalnız siz və server şifrələyib de-şifrələməyi bacarır.

Azərbaycan dilində hərflər tezliyi, Azərbaycan dilində nümunə mətnlər toplusunu istifa-

də etməklə hesablanmışdır.

Şəkil 13.3: Sezar şifrələməsi ilə açarı "3" götürməklə serverə şifrələnmiş məlumatın göndərilməsi.

Əslində heç də pis ideyaya oxşamır. Eyni qayda ilə server də sorğuya cavabı şifrələyib göndərə bilər. Məsələn,

Şəkil 13.4: Sezar şifrələməsi ilə açarı "3" götürməklə serverdən şifrələnmiş məlumatın əldə edilməsi.

İndi təsəvvür edin ki, sniffer vasitəsilə bir haker məlumatı əldə etmək istəyir. O məlumatı əldə edir, lakin məlumat şifrələnmişdir. Açarı bilmədiyindən düşünə bilərsiniz ki, o məlumatı oxuya bilməyəcək.

Şəkil 13.5: Haker şifrmətni ələ keçirir.

Əvəzetmə şifri ilə şifrələnmiş mətni deşifrələmək üçün "brute-force" üsulundan istifadə etmək olar. Bu yolla haker bütün mümkün açarları sınayıb nəticədə orijinal mətni əldə edə bilər.

Şəkil 13.6: Əvəzetmə şifrələməsinin “brute-force” üsulu ilə sındırılması.

Gördüyünüz kimi 12 addımda şifrmətn sındırıldı və haker orijinal mətni əldə edə bildi. Kompüter üçün 12 addım və ya hətta 12 min addım çox qısa zaman kəsimində yerinə yetirilə bilər. Ona görə də bu cür şifrələmə metodları etibarsızdır. Lakin sadə əvəzetmə şifrləri kriptoqrafiyanı öyrənmək və qavramaq üçün vacibdir.

Simmetrik və assimmetrik şifrələmə

Məlumatı göndərən və qəbul edən eyni açarı istifadə etdikdə onlar simmetrik-açar kriptoqrafiyasını tətbiq edirlər. Hansı açarı istifadə etdiklərinə əvvəlcədən razılaşmalıdırlar. Əvvəlki bölmədə sadə simmetrik açara nümunə göstərdik. Simmetrik-açar kriptoqrafiyasının işləyə bilməsi üçün həm göndərən həm də qəbul edən tərəf açar barədə əvvəlcədən razılığa gəlməlidir. Bu məlumatları əvvəlcədən razılaşdırmaq nadir hallarda səmərəli olur. Çünki o açar ələ keçərsə bütün məlumatları deşifrələmək mümkün olacaq. Lakin simmetrik şifrələmə kifayət qədər sürətli olduğundan bir çox hallarda istifadə edilir. Amma bu zaman açarlar adətən qısamüddətli olur. Bu xüsusiyyətinə görə bəzən onlara sessiya açarları da deyirlər.

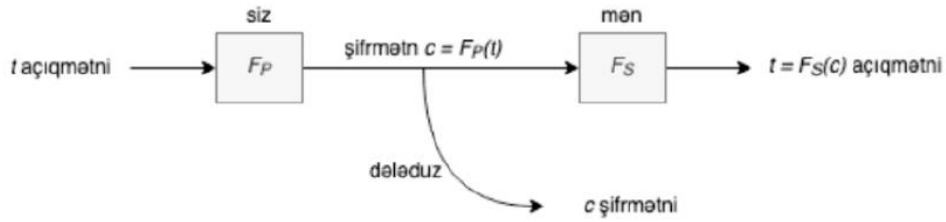
Şübhəsiz ki, şifrələnmiş mesajı qəbul edən şəxsin mesajı deşifrələyə bilməsi üçün həm göndərənin həm də qəbul edən tərəfin şifrələmə açarını əvəəlcədən bilməsi vacibdir. Elədirmi? Əslində belə deyil.

Açıq açar kriptoqrafiyasında hər bir tərəfin iki açarı var: açıq açar və gizli açar. İndi mən iki tərəf arasında açıq-açar kriptoqrafiyasını izah etməyə çalışacağam. Öz açıq-açarımı P ilə, gizli açarımı isə S ilə işarə edəcəyəm. Qarşı tərəfin isə özünün ayrıca açıq və gizli açarları olur. Əgər başqa tərəflər də prosesdə iştirak edərlərsə onların da öz açıq və gizli açar cütlərinin olması gərəkdir.

Gizli açarlar adından göründüyü kimi məxfidir, lakin açıq açarlar hər kəsə məlum ola bilər. Açıq açarlar hətta ixtiyari şəxsin istifadə edə biləcəyi mərkəzləşmiş bir kataloqda da mövcud ola bilər. Müvafiq şərtlər daxilində biz öz aramızda şifrələmə və deşifrələmə üçün bu açarlardan birini istifadə edə bilərik. “Müvafiq şərtlər” dedikdə açıq və gizli açarların ya açıqmətni şifrələmək üçün, ya da şifrmətni deşifrələmək üçün istifadə edən funksiyaların mövcud olduğunu nəzərdə tuturam. Açıq açar ilə istifadə etdiyim funksiyanı FP, gizli açarla istifadə etdiyimi isə FS ilə işarə etdim. Açıq və gizli açar arasında xüsusi qanunauyğunluq növbəti şəkildədir: t = FS(FP (t)).

Bu qaunauyğunluq imkan verir ki, siz mənim açıq açarımla açıqmətni şifrələyə biləsiniz və mən də öz gizli açarımla həmin şifrmətni deşifrələyib orijinal mətni əldə edə bilim. Açıq-açar kriptoqrafiyasının bəzi tətbiq sahələri də var ki, orada qanunayğunluq t = FP (FS (t)) şəklindədir. Onda mən öz gizli açarılma açıqmətni şifrələdikdə istənilən şəxs onu açıq açarla deşifrələyə bilər. Hər kəs mənim açıq açar funksiyam olan FP-ni səmərəli şəkildə hesablaya bilməli, ancaq yalnız özüm gizli açar funksiyam FS-i istənilən məqamda hesablaya bilməliyəm. Eyni qayda bütün tərəflərin açıq və gizli açarları üçün də keçərlidir: FP açıq açar funksiyası asanlıqla hesablanır, ancaq gizli açarla çalışan FS funksiyası

yalnız sahibi tərəfindən hesablana bilər. Siz mənə mesaj göndərdikdə ümumi sxem belə olacaq:



Şəkil 13.7: Assimetrik şifrələmə

Siz mənə şirfələnmiş mesaj göndərmək istədikdə t açıqmətni ilə başlayırsınız. Mənim P açıq açarımı əldə edirsiniz (şəxsən mənim özümdən soruşmaqla, yaxud ümumi kataloqdan tapmaqla). P açarını əldə etdikdən sonra asanlıqla c = FP(t) düsturunu istifadə edib şifrmətni hesablayırsınız. Siz mənə yalnız şifrmətni göndərdiyiniz üçün hansısa dələduz mesajı ələ keçirərsə o yalnız şifrmətni görə bilər. Mən öz növbəmdə c şifrmətnini əldə etdikdən sonra gizli açarımı istifadə edərək mesajı t = FS(c) funksiyası ilə deşifrələyib oxumağa müvəffəq oluram. Siz və ya başqaları mənim açıq açarımı istifadə edərək şifrmətni sürətli şəkildə hesablaya bilərsiniz. Lakin, yalnız mən özüm şifrmətni deşifrələyə bilərəm.

Biz FP və FS funksiya cütlərinin birgə düzgün işləməsindən əmin olmalıyıq. FP-nin hər bir açıqmətn üçün fərqli şifrmətn istehsal etməsini istəyirik. Fərz edək ki, FP funksiyası iki fərqli t1 və t2 açıqmətni üçün eyni şifrmətn ilə nəticələnir. Yəni FP(t1) = FP (t2). Mən FP (t1) şifrmətnini aldıqda və onu FS funksiyasının köməyi ilə deşifrə etdikdə t1-i yoxsa t2-i əldə edəcəyimi bilməyəcəm. Şifrələmənin təsadüfi bir element daxil etməklə baş verməsi yaxşı olardı. Belə ki, eyni açıqmətn iki dəfə FP funksiyasına ötürüldükdə hər dəfə fərqli şifrmətnlərlə nəticələnsin. Məsələn, RSA alqoritmi belə işləyir. RSA sistemində açıqmətn şifrələnən məlumatın yalnız kiçik bir hissəsini təşkil edir. Məlumatın qalan hissəsi boşluqları doldurmaq üçün təsadüfi seçilmiş məlumatlardan ibarətdir. Belə üsul şifrələmə alqoritmini daha etibarlı edir. Əlbəttə ki, şifrənin açılması üçün FS funksiyası qeyd etdiyimiz bütün məqamları nəzərə almalıdır ki, eyni mətn fərqli şifrmətnlərə çevrilmiş olsalar da onu geriyə çevirdikdə eyni açıqmətn əldə edilsin.

Ədəbiyyatlarda və bir sıra proqram təminatlarında rast gələcəyiniz DES, 3DES, AES simmetrik-açarlı kriptoqrafik alqoritmlərə nümunədir. RSA, DSA, Diffie-Helman alqoritmləri isə assimmetrik-açarlı alqoritmlərdəndir.

Növbəti mövzuda HTTPS protokolunda bu üsulları istifadə etməklə müştəri ilə server arasında rabitənin şifrələnməsi üsulunu öyrənəcəyik.

Belə ssenari məlumatın məxfiliyi üçün deyil, onun tamlığına, yəni təhrif olunmadığına əmin olmaq üçün istifadə edilir.

Hash funksiyalar. MD5, SHA1.

Kriptoqrafiyadan söz açmışkən hash funksiyalar barədə də danışmalıyıq. Hash funksiyaları kriptoqrafiya və kompüter elmində əsas anlayışdır və verilənlərin tamlığını yoxlamaqda və autentifikasiyada mühüm rol oynayır. Hash funksiyası istənilən ölçülü giriş verilənlərini sabit uzunluqlu simvollar sətirinə çevirən riyazi alqoritmdir. Hash funksiyasından alınan nəticə və ya qısa olaraq hash hər bir giriş məlumatı üçün unikaldır. Məsələn, MD5 hash funksiyası ilə "salam" sözünün nəticəsi aşağıdakı kimi olacaqdır.

```
salam -> de6838252f95d3b9e803b28df33b4baa
```

Veb təhlükəsizlikdə hash funksiyalar daha çox məlumatların tamlığını təmin etmək və verilənlər bazalarında şifrələrin saxlanması üçün istifadə edilir. Təsəvvür edin ki, istifadəçilərin şifrələri bir sistemin verilənlər bazasında açıq şəkildə yerləşdirilib. Həmin verilənlər bazası kiminsə əlinə keçərsə o şifrələri istifadə edərək sizin hesaba daxil ola bilərlər. Və qorxulusu ondan ibarətdir ki, biz bir çox hallarda eyni şifrəni fərqli resurslar üçün istifadə edirik. Bu o deməkdir ki, şifrəni ələ keçirməklə bizim bir çox hesablarımızı ələ keçirə bilərlər. Bunun qarşısını almaq üçün şifrələri verilənlər bazasına yerləşdirərkən hansısa formada şifrələmək lazımdır. Biz şifrələmə alqoritmlərindən birini istifadə edərək onları şifrələyə bilərik, lakin o zaman da açarı kimsə ələ keçirərsə bütün bazanı deşifrələyə bilər və yuxarıda qeyd etdiyimiz problemlə yenə də qarşılaşarıq.

Əslində biz istifadəçinin şifrəsini bazada elə saxlaya bilərik ki, yalnız istifadəçi hesabına daxil olmaq istəyən zaman onu eyniləşdirə bilək. Yəni eyniliyini yoxlaya bilək. Bunu hash funksiyanın köməyi ilə etmək olar. Məsələn, şifrə "salam" mətnidirsə verilənlər bazasında "de6838252f95d3b9e803b28df33b4baa" sətirini saxlamaq kifayətdir.

```
salam -> de6838252f95d3b9e803b28df33b4baa
```

İstifadəçi hesabına daxil olarkən öz şifrəsini veb saytın müvafiq xanasında daxil edir və sistem eyni alqoritmlə onun hash dəyərini əldə edib verilənlər bazasındakı ilə yoxlayır. Əgər eynidirsə istidaçi login ola bilir.

Sənayədə istifadə edilən bir sıra hash funksiyalar vardır. Onlardan MD5 və SHA1 alqoritmi ilə tanış olaq.

MD5 geniş istifadə olunan kriptoqrafik hash funksiyasıdır və 32 simvoldan ibarət onaltılıq hash dəyəri yaradır. 1991-ci ildə Ronald Rivest tərəfindən hazırlanmış MD5 əvvəlcə təhlükəsiz rəqəmsal imza proqramları və məlumatların bütövlüyünün yoxlanılması üçün nəzərdə tutulmuşdur. Bununla belə, zəifliklər səbəbindən zamanla onun istifadəsi azalıb. Sonralar MD6 versiyası çıxsa da o qədər də geniş istifadə edilmir.

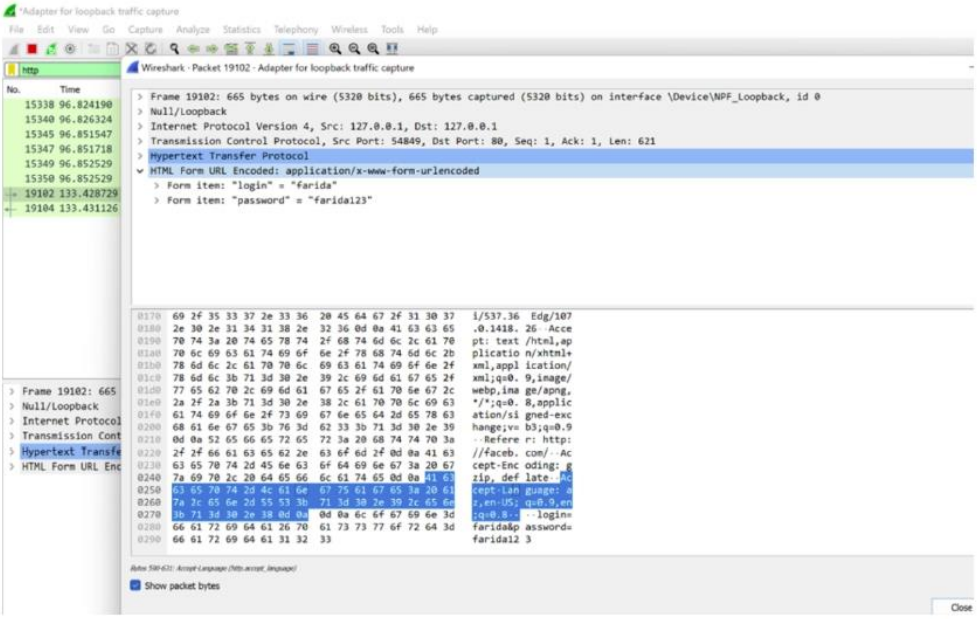
SHA-1 ABŞ Milli Təhlükəsizlik Agentliyi (NSA) tərəfindən hazırlanmış və 1995-ci ildə Milli Standartlar və Texnologiya İnstitutu (NIST) tərəfindən nəşr edilmiş digər kriptoqrafik hash funksiyasıdır. O, 40 simvoldan ibarət onaltılıq heş dəyəri yaradır. Təəssüf ki, SHA-1 alqoritmi də sonradan sındırılmışdır. Sonralar SHA-2 və SHA-3 versiyaları yaradılmışdır.

Bu iki alqoritm sındırılsa da hazırda kifayət qədər geniş istifadə edilir.

SSL

HTTP-nin problemləri

Bu bölmədə veb təhlükəsizliyinin vacib komponenti olan HTTPS-i öyrənəcəyik. HTTPS protokolunun əhəmiyyətini anlamaq üçün ilk olaraq HTTP-yə xas zəifliklər başa düşməyə çalışaq. Daha sonra isə HTTPS-in bu problemləri necə həll etdiyini öyrənək. İnternetdə məlumat kommunikasiyasının əsası olan HTTP-də əhəmiyyətli təhlükəsizlik qüsurları var. HTTP protokolu ilə ötürülən məlumatlar şifrələnmir. Sniffer dərsində öyrəndik ki, şəbəkəni ələ keçirməklə bütün ötürülən məlumatı izləmək mümkündür.



Şəkil 13.8: Wireshark aləti ilə şəbəkədə ötürülən məlumatın ələ keçirilməsi

Bu problemi aradan qaldırmaq üçün ötürülən məlumat şifrələnməlidir. Bəs hansı alqoritmlə? Əvvəlki mövzuda sadə əvəzetmə şifrlərinin etibarsız olduğunu aydınlaşdırdıq. Deməli HTTP ilə ötürülən məlumatları həmin üsullarla şifrələmək olmaz. Məsələn, kredit kart məlumatlarını bəsit alqoritmlə şifrələsək, hakerlər onu asanlıqla ələ keçirə bilər.

O zaman simmetrik açarlı etibarlı kriptoqrafik üsullardan istifadə etmək pis fikir deyil. Amma burada başqa bir problem ortaya çıxır. Bəs tərəflər açarları necə paylaşmalıdırlar? Və o açar ələ keçərsə alqoritmin etibarlılığı heç bir məna kəsb etmir.

Bu problemi həll etmək üçün HTTPS protokolu mövcuddur. O HTTP protokoluna uyğun şəkildə çalışır, lakin SSL (Secure Sockets Layer) adlı şifrələmə üsulunu istifadə edir. SSL ilk dəfə 1995-ci ildə Netscape şirkəti tərəfindən yaradılmışdır. Sonralar onun yeni versiyası çıxmış və TLS (Transport Layer Security) adlandırılmışdır.

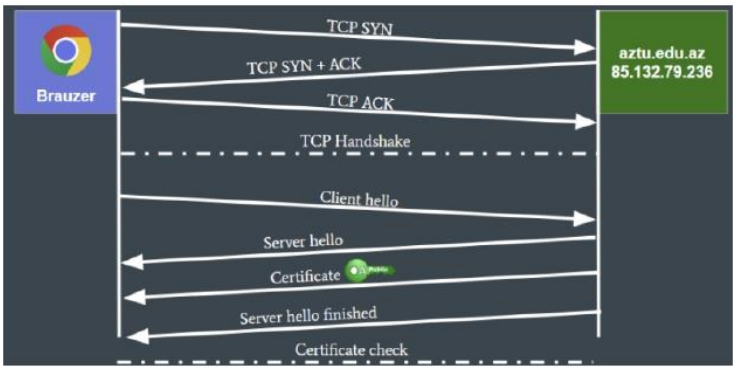
HTTP necə işləyir?

HTTPS və ya təhlükəsiz HTTP, SSL/TLS protokollarının tətbiqi vasitəsilə ötürülən məlumatın təhlükəsizliyini faktiki olaraq artırır. HTTPS ilə brauzer və server arasında ötürülən məlumatların şifrlənir və məlumatın konfidensiallığı təmin edilir. Məlumatın təhriflərdən qorunması da HTTPS protokolunda təmin edilir, beləliklə ötürülən məlumatın tamlığı da qorunmuş olur. Bundan başqa istifadəçilərin saxta vebsaytlara qoşulmasının qarşısını almaq üçün serverin həqiqiliyi yoxlanılır. Bu üç amil HTTPS protokolu vasitəsilə internetdə təhlükəsiz "sayəhət" etməyi təmin edir.

HTTPS protokolu dedikdə SSL/TLS vasitəsilə həyata keçirilən HTTP qoşulması nəzərdə tutulur. Ona görə də HTTPS, SSL və TLS terminləri bir çox hallarda bir-birinin əvəzi kimi də istifadə olunur. Həm mühazirələrdə həm də digər ədəbiyyatlarda bu üç termin bir çox hallarda eyni şeyi bildirir. Bu sizi çaşdırmasın.

İndi gəlin HTTPS-in necə çalışdığını öyrənək. İlk olaraq müştəri ilə server arasında şəbəkə bağlantısı qurulmalıdır. Bunun üçün TCP bağlantısı müştəri tərəfindən başladılır. Yəni, SYN ("sinxronizasiya") konfigurasiyası ilə TCP paketi yaradılıb serverə göndərilir. Server paketi qəbul edir və öz yaddaşında TBC (Transmission Control Block) yaradır və həm SYN, həm də ACK konfiqurasiya bayraqcıqlarını (ingilicsə, flag) əlavə edib TCP paketi ilə müştəriyə cavab verir. Müştəri ACK bayrağı dəsti ilə bir daha TCP paketi göndərərək serverə cavab verir. Buna üçayaqlı "əlsıxma" (ingiliscə, three-way handshake) prosesi də deyirlər. Bu üç addımdan sonra istifadəçi ilə server arasında bağlantı əmələ gəlir və onlar bu bağlantı ilə növbəti məlumatları bir-birinə göndərə bilərlər. Bu prosesə "TCP handshake" və ya "TCP əlsıxması" deyilir.

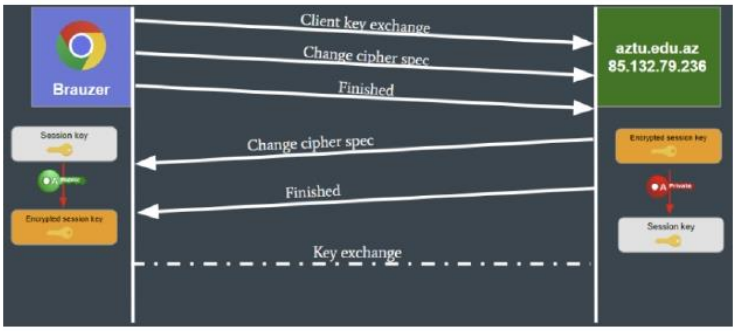
Bağlantı əmələ gəldikdən sonra müştəri (istifadəçi) serverə "Client hello"mesajını göndərərək serveri "salamlayı". Mesaj bir sıra zəruri şifrələmə alqoritmlərini (şifrə dəstləri) və dəstəkləyə biləcəyi ən son TLS versiyasını ehtiva edir. Yəni müştəri (məsələn, brauzer) öz dəstəklədiyi şifrələmə alqoritmlərini serverə bildirir. Server "Server hello" mesajı ilə cavab verir və brauzerə onda olan alqoritmləri və TLS versiyasını dəstəkləyə biləcəyini bildirir. Və dərhal server müştəriyə ikinici mesajı, SSL sertifikatını göndərir. Sertifikatda açıq açar, host adı, sertifikatın son istifadə tarixləri və s. məlumatlar olur. Bununla da sertifikatın yoxlama prosesi bitmiş olur.



Şəkil 13.9: SSL üçün serverlə bağlantının qurulması mexanizmi

Bu addımdan sonra müştəri serverin açıq açarını bilir, yəni onu istifadə edərək məlumatı şifrələyib serverə göndərə bilər. Server öz gizli açarı ilə o məlumatı deşifrələyib oxuya biləcək. Bu prosesin detallarını mühazirənin əvvəlində izah etmişdik. Bəs server müştəriyə məlumatı hansı açarla şifrələyib göndərməlidir? Məntiqlə, eyni prinsip ilə müştəri də bir açar generasiya edib onu serverə göndərə bilər. Beləliklə server müştəriyə cavab göndərərkən müştərinin açıq açarı ilə məlumatı şifrələyər. SSL protokolunda bu məqam bir qədər fərqli işləyir. Çünki assimmetrik şifrələmə çox resurs istifadə edir. Belə olduqda şifrələmə və deşifrələməyə vaxt çox gedərdi.

Əslində SSL vasitəsilə məlumat mübadiləsi simmetrik açarlı şifrələmə ilə baş tutur. Yuxarıda təsvir etdiyimiz addımlardan sonra müştəri bir sessiya açarı generasiya edir və o açarı serverin açıq açarı ilə şifrələyib serverə göndərir. Server onu deşifrələyir.

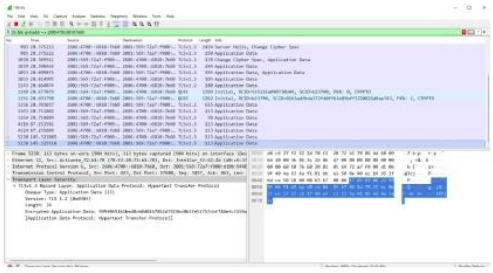


Şəkil 13.10: SSL üçün sessiya açarının generasiyası və serverə ötürülməsi.

Bu andan etibarən həm server həm də müştəri eyni açara malik olur və bundan sonra onlar öz aralarında simmetrik açarlı şifrələmə ilə məlumat mübadiləsi həyata keçirirlər.

Şəkil 13.11: SSL ilə məlumat mübadiləsi.

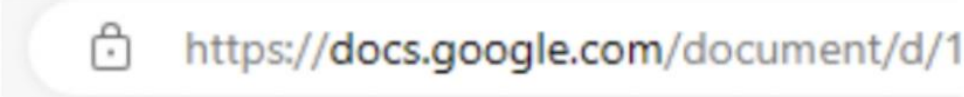
HTTPS ilə ötürülən məlumatı sniffer ilə əldə etmək istəsək şifrələnmiş məlumatı görə biləcəyik. Onu deşifrələyə bilməyəcəyik.



Şəkil 13.12: HTTPS ilə ötürülən məlumata sniffer vasitəsilə baxış

İnanılmış (Trusted) sertifikat mərkəzləri

SSL ilə qorunan hər hansı bir səhifəyə daxil olduqda brauzer onun həqiqiliyini necə müəyyənləşdirə bilər? Brazuerlərin daxilindəki verilənlər bazasında etibarlı sertifikat mərkəzlərinin siyahıları və sertifikatları mövcuddur. Yalnız bu siyahıdakı sertifikat mərkəzinin verdiyi sertifikat ilə qorunan saytlara daxil olduqda brauzer sizə icazə verir və ünvanlar panelində qıfıl şəklini görürsünüz.



Şəkil 13.13: Etibarlı sertifikatı bildirən qıfıl şəkli

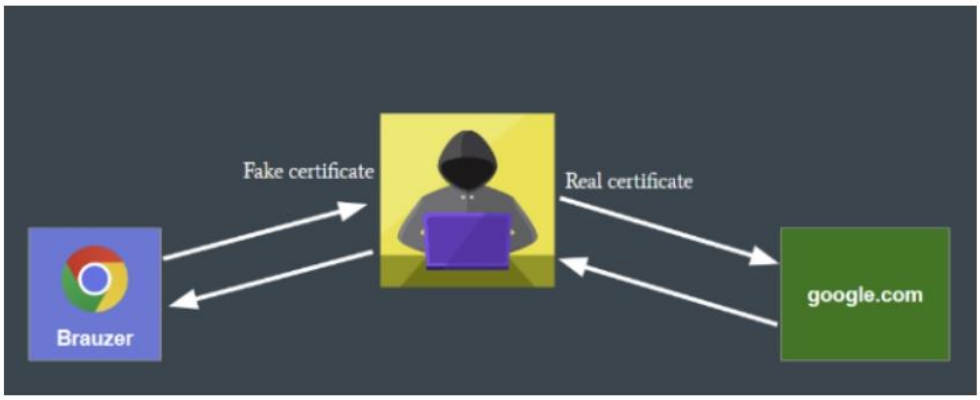
Brauzer proqramları mütəmadi şəkildə bu bazaları yeniləyir və brazuerin yenilənməsi zamanı yeni sertifikatlar da onun daxilində kompüterinizə köçürülür. Bu sertifikat mərkəzləri kifayət qədər ciddi qorunan və nüfuza malik qurumlar tərəfindən yaradılmışdır. Bu siyahıya düşmək üçün sertifikat mərkəzləri ciddi şəkildə hazırlaşmalıdırlar. Bu mühazirə yazılan tarixdə (2024-cü il Mart ayı) Azərbaycanın heç bir sertifikat mərkəzi həmin

siyahıya təəssüf ki, daxil olmamışdır.

HTTPS tam təhlükəsizdir?

HTTPS-in köməyi ilə mübadilə olunan məlumatların şifrələnməsini və təhlükəsizliyin təmin olunmasını müzakirə etdik. Bəs həqiqətən də HTTPS təhlükəsizliyə tam zəmanət verirmi?

Rəqəmsal təhlükəsizlik tarixində ən diqqətəlayiq pozuntulardan biri 2011-ci ildə Hollandiyanın sertifikat mərkəzi orqanı olan DigiNotar ilə baş vermişdi. DigiNotar 2011-ci ilin iyulunda hakerlər tərəfindən ələ keçirilmiş və sertifikat mərkəzinin adından bir çox məşhur saytlara – Google.com, Gmail.com və s. sertifikatlar yaradılmışdı. Daha sonra şəbəkələrə müdaxilə edərək istifadəçilərdən gələn sorğuları öz serverlərinə yönləndirmişdilər. Məsələ burasındadır ki, DigiNotar etibarlı sertifikat mərkəzlərindən olduğundan bütün brauzerlər bu sertifikatların həqiqiliyini tanıyır və istifadəçilər heç nədən xəbər tuta bilməmişdi. Əslində isə istifadəçilərin bütün məlumatları dələduzların serverlərindən keçirmiş. Məlumatlar dələduzların serverinə DigiNotarın sertifikatı ilə HTTPS protokolu vasitəsilə gedir, dələduzlar məlumatları oxuyur və Google-un serverlərinə həqiqi sertifikatla göndərirdilər. Yəni Man-in-the-middle-attack həyata keçirmişdilər. Bu hadisə zamanı 300 000-dən çox İran vətəndaşının şəxsi məlumatları oğurlanmışdı.



Şəkil 13.14: SSL üzərindən Man-in-the-middle-attack

Bu hadisədən sonra DigiNotar şirkəti etibarlı sertifikat mərkəzləri siyahısında çıxarıldı və şirkət elə həmin il müflis oldu. Bu hadisə onu göstərir ki, HTTPS protokolunun iş mexanizmi nə qədər də təhlükəsiz olsa da heç də 100 təhlükəsizlik mənasına gəlməməlidir.

14 | Mühazirə 14

Autentifikasiya

Autentifikasiya nədir?

Təsəvvür edin bir müəssisəyə getmisiniz. Orada girişdə sizdən şəxsiyyət vəsiqəsinizi və ya şəxsiyyətinizi təsdiq edən hər hansı bir sənəd istəyirlər. Siz sənədi təqdim edirsiniz və onları sənəddə şəkil ilə sizin eyni adam olduğunuzu gözəyarı vizual olaraq müəyyənləşdirir. Əgər eyni adamdırsa onda sizi içəri buraxırlar.

Bəzi şirkətlərin binasına daxil olduqda şirkətin işçiləri öz şirkət vəsiqələrini qapıdakı qurğuya yaxınlaşdırır və şirkət əməkdaşının kartı aktivdirsə qapılar açılır. Bəzi şirkətlərdə isə üz tanıma sistemi ilə işçini müəyyən edən qurğular vardır və qapılar avtomatik açılır.

Bu deyilənlər hamısı autentifikasiyaya nümunədir. Autentifikasiya dedikdə bir şəxsin hər hansı bir yerə girişi üçün hansısa məlumatların doğrulanması nəzərdə tutulur. Kompüter aləmində və ya kiber aləmdə bu istifadəçilərin bir sistemə daxil olması və ya bir sistemin digər sistemə daxil ola bilməsi başa düşülür.

Autentifikasiya kompüter sistemindəki istifadəçinin, cihazın və ya digər iştirakçının (sistem, təşkilat və s.) kim olduğunun yoxlanılmasıdır. Adətən həmin sistemdəki resurslara girişə icazə vermək üçün ilkin giriş nöqtəsidir. NİST-in lüğətinə istinad etsək “Adətən informasiya sisteminə girişin ilkin şərti olaraq bir istifadəçinin, prosesin və ya avadanlığın kimliyinin müəyyən edilməsi.” deməkdir.

Autentifikasiyanın növləri.

Autentifikasiya sistemləri adətən aşağıdakı növlərə bölünür: Sizin bildiyiniz bir şey - Something you know (məsələn, şifrə, pin kod və s.) Sizdə olan bir şey - Something you have (məsələn, smartfon, kriptoqrafik açar, badge və s.) Sizin özünüz - Something you are (məsələn, barmaq izi, göz qişası, və s.)

Sizin bildiyiniz bir şey dedikdə hansısa məxfi şifrədən söhbət gedir. Məsələn, hər hansı bir saytda login səhifəsində siz istifadəçi adı və şifrəinizi daxil edərək sistemə giriş əldə edirsiniz. Bu “sizin bildiyiniz bir şey”ə nümunədir.

Buna başqa bir nümunə kredit kartınızın PİN kodudur. Məsələn, siz alış-veriş edərkən kartınızı POS terminala daxil etdikdə və ya ATM avadanlığına daxil etdikdə PİN kodu daxil etməlisiniz.

Sizdə olan bir şey dedikdə autentifikasiya üçün sizdə fiziki olaraq olan bir şeyin istifadə olunması nəzərdə tutulur. Məsələn, aşağıdakı şəkildə RSA SecurID tokeni göstərilmişdir.

Bəzi şirkətlər öz işçilərinin şirkətin sistemlərinə daxil olması üçün bu tokenlərdən istifadə edir. Bu tokenin üzərində olan ədədlər vaxtaşırı (1 dəqiqədən bir, 30 saniyədən bir və s.) dəyişir. Bu cür tokenlər adi şifrələrdən daha təhlükəsizdir. Şifrəni ələ keçirdikdə dələduzlar onu həmişə istifadə edə biləcək. Amma bu tokeni ələ keçirsələr belə ciddi nəsə edə

bilmələri riski çox aşağıdır. Çünki bir neçə dəqiqədən sonra həmin token dəyişəcək və onların əldə etdikləri token çalışmayacaq. Təbii ki, burada token qurğusunun ələ keçirilməsi aid deyil.

Müasir dövrdə bu cür qurğunu smartfonlarda olan proqramlar da əvəz edə bilir. Microsoft Authenticator, Google Authenticator ən geniş yayılmış nümunələrdir. Yuxarıdakı token qurğularından başqa kompüterə qoşulan fərqli formada autentifikasiya qurğuları da mövcuddur. Məsələn, şəkildə gördüyünüz YubiKey adlı qurğu kompüterin USB portuna qoşulur. Siz hər hansı bir sistemə daxil olarkən istifadəçi adı və şifrədən başqa uzun bir kod da istənilir. O kod daim dəyişir. Həmin bu YubiKey qurğusuna barmağınızı sürtdüyünüz zaman o kod generasiya edilir və lazım olan xanaya sanki klaviaturadan yazılırmış kimi yazılır.

Sizin özünüz dedikdə autentifikasiya sistemləri konteksində biometrik məlumatlar başa düşülür. Burada bir sistemi daxil olmaq üçün sizin özünüzün istifadəsi nəzərdə tutulur. Məsələn, barmaq izi, göz qişası, sifət və s. bunlar biometrik məlumatlardır.

Müasir smartfonların çoxunda telefonu blokdan çıxarmaq üçün sadəcə telefona baxmaq kifayətdir. Telefon sizin göz qişanızı və ya sifətinizi skan edir və sizin olub-olmadığınızı müəyyən edir.

İnsanın səsi də biometrik məlumat olduğu üçün bəzi banklarda müştəri xidmətlərinə zəng edən adamın həqiqətən siz olub-olmadığınızı yoxlamaq üçün səs tanıma sistemindən istifadə edirlər.

Bunlardan başqa autentifikasiya üsulları da mövcuddur. Məsələn, məkan faktoru və ya insanın davranış faktoru da autentifikasiya məlumatı kimi istifadə edilə bilər. Bəzən giriş cəhdinin edildiyi yer identifikasiya faktoru kimi xidmət edə bilər, məsələn, təhlükəsiz fiziki məkandan və ya müəyyən IP ünvan diapazonundan xidmətə daxil olmaq. Bu üsul xüsusilə korporativ şəbəkələrdə geniş yayılmışdır. Hər-hansı bir sistemə daxil olarkən istifadəçinin İP ünvanı yoxlanılır və sistemə girişə icazə İP ünvana əsasən verilir.

Klaviaturada daxil edilən düymələrin vuruş dinamikası, mausun hərəkətləri və hətta yerimə nümunələri kimi davranış biometrik məlumatlar da eyniləmə üçün istifadə edilə bilər. Autentifikasiya sistemlərində geniş yayılmasa da insanların müəyyən edilməsi üçün internetdə izləmə texnologiyalarında bu üsullar geniş istifadə edilir.

Praktikada autentifikasiya

Autentifikasiyanın ümumi olaraq nə olduğunu və onun növlərini əvvəlki bölmədə öyrəndik. İndi isə praktikada veb sistemlərdə autentifikasiyanın necə çalışdığını, bunun üçün hansı texnologiyaların istifadə edildiyini araşdıraq. Bunun üçün gəlin praktiki nümunədən istifadə edərək onlayn alış-veriş platforması kimi tipik veb proqramda autentifikasiyanın necə işlədiyinə ətraflı nəzər salaq. Hər mərhələdə iştirak edən texnologiyaları və sistemləri vurğulayaraq, prosesi addım-addım nəzərdən keçirəcəyik. Nümunə kimi bir alış-veriş platformasına baxaq.

İstifadəçi onlayn alış-veriş platformasında öz hesabına daxil olmaq istədikdə, onlara veb saytda giriş forması təqdim olunur. Bu forma, adətən HTML, CSS və JavaScript ilə qurulan, istifadəçinin brauzerində işləyən istifadəçi interfeysinin bir hissəsidir. Forma istifadəçinin istifadəçi adı və şifrə tələb edir.

İstifadəçi adı və şifrə daxil edildikdən sonra bunlar brauzerdən serverə göndərilir. Məlumatlar ötürülərkən onları qorumaq üçün məlumat brauzer və server arasında rabitə kanalını şifrələmək üçün SSL/TLS-dən istifadə edən HTTP-nin təhlükəsiz versiyası olan HTTPS-dən istifadə etməklə şifrələnir. SSL və HTTPS barədə əvvəlki mühazirələrdə söhbət açmışdıq və qeyd etmişdik ki, HTTPS protokolu internet üzərindən göndərilən məlumatların təhlükəsiz olmasını və zərərli şəxslər tərəfindən asanlıqla ələ keçirilməməsini təmin edir.

Server giriş sorğusunu API vasitəsilə qəbul edir. Node.js, Python, Java və ya bu kimi digər texnologiyalarla yaradılmış server proqramı istifadəçi adı və şifrəni qəbul edir və onları verilənlər bazası ilə yoxlayır. İstifadəçinin təqdim etdiyi şifrə kriptoqrafik heşinq alqoritmindən istifadə etməklə heşlənir və bu hash verilənlər bazasında saxlanılan heş ilə müqayisə edilir. Heş barədə də kriptoqrafiyaya aid mühazirə də bəhs etmişdik.

İstifadəçinin daxil etdiyi istifadəçi adı və şifrə MySQL və ya PostgreSQL kimi SQL verilənlər bazası və ya MongoDB kimi NoSQL verilənlər bazası ola bilən verilənlər bazasında saxlanılır. Verilənlər bazası identifikasiya üçün server tərəfindən alınan istifadəçi adı

və şifrələnmiş şifrəni saxlayır. Bəzi autentifikasiya sistemləri Microsoft Active Directory əsasında çalışır. Onun özünün verilənlər bazası vardır.

Uğurlu autentifikasiyadan sonra server JWT (JSON Web Token) adlı bir data yaradır. Bu data server tərəfindən kodlaşdırılan və imzalanan istifadəçi təfərrüatları və icazələri olan məlumatlardan ibarətdir. Token daha sonra istifadəçinin brauzerinə göndərilir, burada onu yaddaşda (local storage) və ya kukilərdə saxlamaq olar.

Müştəri tərəfi resurslara daxil olmaq üçün JWT tokenini hər bir HTTP sorğusunun başlıqlarına daxil edir. Bu token istifadəçinin autentifikasiyasını uğurla həyata keçirməsinə sübut kimi xidmət edir və istifadəçiyə hər yeni səhifəyə yenidən daxil olmağa ehtiyac qalmadan proqramla əlaqə saxlamağa imkan verir. Server tərəfi həmin tokenin həqiqiliyini tokendəki imza ilə yoxlaya bilir. JWT ilə olan üsul yeganə üsul deyil. Bəzi üsullarda sessiya açarları kukilərdə saxlanılır və server həmin kukilərə əsasən istifadəçinin həqiqiliyini yoxlaya bilir. Sessiyalar və kukilər barədə əvvəlkə mühazirələrdə danışmışdıq.

Gördüyünüz kimi sadə bir onlayn alış-veriş platforması üçün autentifikasiya veb tətbiqin fərqli elementlərindən tutmuş təhlükəsizlik protokollarına və verilənlər bazası sistemlərinə qədər müxtəlif texnologiyaları istifadə edən bir prosesdir.

Serverlərarası autentifikasiya

Biz indiyədək istifadəçi ilə server arasındakı autentifikasıyadan danışdıq. Müasir dövrdə bir xidmət üçün birdən çox server istifadə edilir. Bu serverlər fiziki və ya virtual serverlər və yaxud konteynerlər ola bilər. Bu servislər də öz aralarında məlumat mübadiləsi aparır. Məsələn, iki mikroservis öz aralarında məlumat mübadiləsi apara bilər. Bu servislər və ya mikroservislər fiziki olaraq fərqli məkanlarda və hətta fərqli ölkələrdə ola bilərlər. Ona görə onların öz aralarındakı data mübadiləsi də təhlükəsiz şəraitdə olmalı və autentifikasiya prosesindən keçməlidirlər.

Serverlərarası autentifikasiya üçün geniş yayılmış metodların bəziləri aşağıdakılardır.

Ortaq Məxfi Açarlar: Serverlər hər bir əməliyyat zamanı imza yaratmaq üçün istifadə edilən məxfi açarlardan istifadə edə bilər. HMAC (Hash-based Message Authentication Code) hər iki serverin məxfi açarı bilməli olduğu ümumi üsuldur və bu açar heç vaxt şəbəkə üzərindən ötürülmür. Açarlar əvvəlcədən hər iki tərəfdə yaddaşda saxlanılır. Məlumat kriptoqrafik üsullarla bu açarları istifadə edərək şifrələnib deşifrələnir. Bu üsul sürətli çalışsa da o qədər də təhlükəsiz deyil. Çünki açarların sızma riski mövcuddur.

SSL/TLS Sertifikatlar: HTTPS haqqında danışdığımız mühazirədə istifadəçi brazueri ilə server arasındakı əlaqədən danışmışdıq. Orada sertifikatların tanınması prosesinə toxunmuşduq. Eyni prinsipi istifadə edərək serverər də öz aralarında autentifikasiya üçün SSL istifadə edə bilər. Secure Sockets Layer (SSL) və ya Transport Layer Security (TLS) kompüter şəbəkəsi üzərindən təhlükəsiz rabitəni təmin edən kriptoqrafik protokollardır. Serverdən serverə rabitədə hər iki server SSL/TLS sertifikatlarından istifadə edərək bir-

birini autentifikasiya edə bilər. Bu quraşdırma hər bir serverin digərinin tanınmış və etibarlı sertifikat orqanlarına qarşı yoxlaya biləcəyi bir sertifikat təqdim etməsini nəzərdə tutur.

OAuth: Veb API-ləri üçün OAuth, bir serverə istifadəçi etimadnamələrini ifşa etmədən digər server tərəfindən yerləşdirilən resurslara daxil olmaq imkanı verən etibarlı giriş üçün istifadə edilə bilər. Bu, müxtəlif xidmətlərin bir-biri ilə etibarlı şəkildə qarşılıqlı əlaqədə olması lazım olduğu mikroservis arxitekturalarında geniş istifadə olunur.

JWT (JSON Veb Tokenləri): JWT-lər tərəflər arasında tokenləri kodlaşdırmaq və təhlükəsiz şəkildə ötürməklə serverdən serverə autentifikasiya üçün istifadə edilə bilər. Bir server xüsusi tokenlərlə JWT yaradır və onu təhlükəsiz üsulla imzalayır. Daha sonra alıcı server əvvəlcədən paylaşılan açar və ya açıq/bağlı açar cütündən istifadə edərək imzanın həqiqiliyini yoxlayır.

Bu siyahı serverlərarası autentifikasiya üçün tam siyahı deyil. Bir çox şirkətlər bu üsullardan fərqli üsullar və ya bir neçə üsulun hibridini də istifadə edirlər.

Serverdən serverə autentifikasiya, dələduzların iki server arasında əlaqəni ələ keçirməsinin qarşısını almalıdır. Bütün kommunikasiyalar üçün şifrlənmiş kanallardan (HTTPS kimi) istifadə etmək, müasir şifrələmə üsullarından istifadə etmək və müntəzəm olaraq fırlanan (key rotation) açar və sertifikatlar istifadə olunan vacib təcrübələrdir. Serverlərarası autentifikasiya bulud hesablamaları, API-ların qarşılıqlı əlaqələri və mikroservis arxitekturaları kimi mühitlərdə olduqca vacibdir.

Növbəti dərsimizdə autentifikasiya zamanı istifadə olunan şifrələrin etibarlılığı barədə danışacağıq.

Şifrələrin etibarlılığı

Şifrələrin sındırılması və etibarlılığı

İstifadəçi şifrəsinin yalnız bir simvoldan ibarət olduğu bir xəyali ssenarini nəzərdən keçirək. Sadəlik üçün yalnız kiçik hərflərin istifadə edildiyini fərz etsək, bu simvol 0-dan 9-a qədər istənilən rəqəm və ya a-dan z-ə qədər hərf ola bilər. Bu şifrəni sındırmaq üçün haker düzgün şifrə tapılana qədər hər bir mümkün rəqəmi və hərfi sınayacaq. Bu, maksimum 10 + 26 = 36 mümkün kombinasiyanı sınamaq deməkdir. Müasir hesablama gücünü nəzərə alsaq, bu proses demək olar ki, heç vaxt aparmayacaq. Saniyənin kiçik bir hissəsi qədər zaman sərf edərək kompüter bu şifrəni dərhal sındıracaq.

İndi bu nümunəni iki simvoldan ibarət şifrə üçün genişləndirək. Şifrənin hər bir simvolu hərf və ya rəqəm ola bilər. İki simvoldan ibarət şifrənin hər mövqeyinin 36 mümkün variantı var (10 rəqəm + 26 hərf), nəticədə 36 x 36 = 1296 mümkün kombinasiya əldə edilir. Bu, daha çox kimi görünsə də, saniyədə minlərlə, hətta milyonlarla təxminləri emal edə bilən müasir kompüterlər üçün hələ də nisbətən kiçik rəqəmdir.

Şifrənin uzunluğunu artırdıqca, mümkün birləşmələrin sayı eksponensial olaraq artır. Üç simvoldan ibarət şifrə

$$36^3$$

(təxminən 46,656) kombinasiyaya gətirib çıxarır. Hər bir əlavə simvol mümkün təxminlərin sayını 36-ya vurur və bu, şifrənın sındırılması üçün lazım olan vaxtın kəskin artmasına səbəb olur.

İndi yalnız rəqəmlər və kiçik hərfləri deyil, həm də böyük hərfləri və @, # $ və s. kimi xüsusi simvolları daxil etməyin faydasını nəzərdən keçirək. Əgər 10 xüsusi simvolun olduğunu fərz etsək, bu, 10 (rəqəm) + 26 (kiçik hərf) + 26 (böyük hərf) + 10 (simvol) = hər mövqe üçün 72 mümkün simvol olacaq. Ona görə də indi tək simvollu şifrənin 72 imkanı, iki simvollu şifrənin isə

$$72^2$$

(5,184) kombinasiyası olacaq. Üç simvoldan ibarət şifrə isə

$$72^3$$

(373,248) kombinasiyaya çatacaq. Təbii ki, böyük rəqəm olsa da müasir kompüterlər üçün bu şifrəni sındırmaq da o qədər də çətin iş deyil.

Simvolların qarışığı olan daha uzun şifrələr mürəkkəbliyi eksponent olaraq artırır. Məsələn, bu tam simvol dəstindən istifadə edən səkkiz simvoldan ibarət şifrənin

$$72^8$$

(təxminən 722 trilyon) mümkün kombinasiyası olacaq. Təsəvvür edə bildiyiniz kimi, bu uzunluqda şifrəni sındırmaq üçün tələb olunan vaxt cari hesablama gücü ilə saatlar alacaq. Simvolların sayını bir az da artırdıqda, məsələn, 12 etdikdə onu sındırmaq üçün 34 min il lazım olar. Bu isə praktiki cəhətdən mümkün deyil.

Şifrənizin etibarlılığı onun uzunluğu artdıqca və istifadə olunan simvolların çeşidi (rəqəmlər, kiçik hərflər, böyük hərflər və xüsusi simvollar) genişləndikcə artmış olur. Şifrə uzun olduqda və istifadə edilən simvollar dəsti çox olduqda onu sındırmaq daha çətin və ya qeyri-mümkün olur. Bu mövzudakı şifrə sındırma metodu bütün mümkün kombinasiyaları yoxlamaqla yerinə yetirilir. Buna "zor güclə" sındırma və ya daha çox bruteforce hücumu deyirlər.

Şifrələrin qorunması

Şifrələri effektiv şəkildə qorumaq üçün həm istifadəçilər, həm də serverlər qabaqlayıcı tədbirlər görməlidirlər. Bu ehtiyat tədbirləri həssas şəxsi və maliyyə məlumatlarını icazəsiz girişdən və potensial təhlükəsizlik pozuntularından qorumaq üçün çox vacibdir.

İstifadəçilər üçün ilk müdafiə xətti güclü şifrələr yaratmaqdır. Bu, böyük və kiçik hərflər, rəqəmlər və xüsusi simvollardan ibarət simvolların birləşməsindən istifadə etməyi nəzərdə tutur. Daha uzun şifrə adətən ən azı 12-16 simvoldan ibarət olması tövsiyə olunur, mürəkkəbliyi əhəmiyyətli dərəcədə artırır və təcavüzkarların kobud güc və ya digər ümumi sındırma üsulları vasitəsilə sındırılmasını çətinləşdirir. Bundan əlavə, istifadəçilər "parol", "123456" və ya "qwerty" kimi asanlıqla təxmin edilən şifrələrdən istifadə etməməlidirlər.

Digər mühüm təcrübə müxtəlif saytlar və xidmətlər üçün unikal şifrələrdən istifadə etməkdir. Bu yolla, bir şifrə oğurlanarsa, digər hesablar təhlükəsiz olaraq qalır. Bundan əlavə, mümkün olan yerdə çox faktorlu autentifikasiyanın (MFA) aktivləşdirilməsi əlavə təhlükəsizlik səviyyəsini artırır. MFA şifrədən başqa əlavə olaraq bir və ya bir neçə

yoxlama metodunu tələb edir, bu, barmaq izi, üz skanı və ya istifadəçinin mobil cihazına göndərilən və ya yaradılan kod ola bilər. Bu kodlara tokenlər deyilir. Bu tokenləri əldə etmək üçün əvvəlki dərsdə danışdığımız YubiKey, RSA SecureID və ya Google Authenticator kimi üsulları istifadə etmək olar.

Şifrələrin qorunması yalnız istifadəçilərin işi deyil. Şifrələr saxlanılan və şifrə tələb edən sistemlər daha məsuliyyətli olmalıdırlar. Server tərəfində saxlanılan istifadəçi şifrələrinin qorunmasını artırmaq üçün bir neçə strategiya var. Birincisi və ilk növbədə şifrələri heç vaxt açıq mətn kimi saxlanmamalıdır. Bunun əvəzinə, şifrələr md5, sha1, bcrypt, scrypt və bu kimi heş alqoritmlərindən istifadə etməklə saxlanmalıdır. Bu alqoritmlər şifrələri asanlıqla geri qaytarıla bilməyən unikal simvollar sırasına çevirir. Şifrələri həm də "duz əlavə etmək" üsulu ilə qarışdırmaq lazımdır.

İstifadəçinin cihazı və server arasında ötürülən məlumatların şifrələnməsi üçün HTTPS-dən istifadə, şifrənin məxfiliyini qorumaq vacibdir. Bundan əlavə, şifrə siyasəti, hesabın bloklanması siyasətlərinin həyata keçirilməsi hesabı müvəqqəti olaraq kilidləməzdən əvvəl icazə verilən uğursuz giriş cəhdlərinin sayını məhdudlaşdırmaqla və ya sonrakı cəhdləri yavaşlatmaqla bruteforce hücumlarını azaltmağa kömək edə bilər.

Avtorizasiya

NİST-in terminologiya lüğətinə əsasən avtorizasiya dedikdə "Hər hansı bir resursu əldə etmək üçün verilən hüquq (səlahiyyət) və ya icazə" başa düşülür. Avtorizasiya sistem resursları, o cümlədən fayllar, xidmətlər, kompüter proqramları və verilənlərlə bağlı istifadəçi imtiyazlarını və ya giriş səviyyələrini müəyyən etmək üçün istifadə edilən təhlükəsizlik mexanizmidir. Bu, uğurlu autentifikasiyadan sonra gələn bir prosesdir; autentifikasiya vasitəsilə istifadəçinin şəxsiyyəti təsdiqləndikdən sonra avtorizasiya həmin istifadəçiyə nə etməyə icazə verildiyini müəyyən edir.

Məsələn, sizin bir Facebook səhifəniz var, orada siz istənilən məlumatı paylaşa bilirsiniz. Dostunuz isə məlumat paylaşdıqdan sonra sizin təsdiqinizi gözləməlidir. Bütün digər istifadəçilər isə yalnız səhifədə siz və ya dostunuz tərəfindən yazılan məlumatları oxuya bilərlər. Beləliklə, sizin post yazmaq üçün bütün səlahiyyətləriniz var, dostunuzun məhdud səlahiyyəti var, digər istifadəçilərin isə daha məhdud. Bu səlahiyyəti müəyyən etmə prosesi avtorizasiya adlanır. Facebookda hansısa avtorizasiya mexanizmi sizə bütün səlahiyyətləri dostunuza isə məhdud səlahiyyətləri verir.

Əsasən, avtorizasiya müxtəlif istifadəçilərə və resurslara giriş hüquqlarını və imtiyazlarını müəyyən etməyi nəzərdə tutur. O, istifadəçinin sistem resursuna daxil ola biləcəyini və ya təşkilat tərəfindən müəyyən edilmiş qaydalar və ya siyasətlər əsasında oxumaq, yazmaq, redaktə etmək və ya silmək kimi əməliyyatları yerinə yetirə biləcəyini müəyyən edir. Bu proses autentifikasiya edilmiş istifadəçilərə sistem və ya şəbəkə daxilində icazə verilən məhdudiyyətləri tətbiq etməyə kömək edir.

Avtorizasiyanın müxtəlif modelləri vardır:

Discretionary Access Control (DAC): Bu modeldə resursun sahibi (məsələn, fayl və ya verilənlər bazası girişi) ona kimin daxil ola biləcəyinə və hansı imtiyazlara malik olduğuna qərar verir. O, "ixtiyari" adlanır, çünki girişə nəzarət sahibinin ixtiyarına əsaslanır.

Mandatory Access Control (MAC): MAC altında resurslara giriş mərkəzi orqan tərəfindən siyasətlərə əsaslanır. O, tez-tez məlumatların yüksək dərəcədə məxfiliyini və bütövlüyünü tələb edən mühitlərdə istifadə olunur.

Role Based Access Control (RBAC): Bu, korporativ mühitlərdə icazənin həyata keçirilməsinin ən ümumi üsullarından biridir. RBAC-da icazələr birbaşa fərdi istifadəçilərə təyin edilmir; daha doğrusu, onlar rollarla əlaqələndirilir və istifadəçilər bu rollara təyin olunurlar. Bu, istifadəçi icazələrinin idarə edilməsini asanlaşdırır və giriş siyasətlərinin ardıcıl şəkildə tətbiq olunmasını təmin etməyi asanlaşdırır.

Həssas məlumatların və kritik sistemlərin təhlükəsizliyini qorumaq üçün müvafiq icazə mexanizmləri çox vacibdir. Onlar icazəsiz girişin qarşısını almağa və hesabın oğurlanması halında baş verə biləcək potensial zərəri məhdudlaşdırmağa kömək edir. İstifadəçilərin yalnız öz iş funksiyalarını yerinə yetirmək üçün lazım olan icazələrə malik olmasını təmin etməklə, təşkilatlar sistemlərin təsadüfi və ya bilərəkdən sui-istifadəsindən qoruya bilər.

Ədəbiyyat

1. Hoffman, Andrew. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. N.p., O'Reilly Media, 2020.
2. Tomas Kormen. Alqoritmlərin Sirri (Azərbaycan). Altun Kitab, 2022.
3. Elie Saad, Rick Mitchell. OWASP Web Security Testing Guide, version 4.2. 2020
4. OWASP Top Ten (https://owasp.org/www-project-top-ten/)
5. Pinto, Marcus, and Stuttard, Dafydd. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. United Kingdom, Wiley, 2011.
6. What is a Browser and How do they work? (https://www.browserstack.com/guide/what-is-browser)
7. How browsers work (https://web.dev/articles/howbrowserswork)
8. https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type
9. `https://www.w3schools.com/html/html_form_input_types.asp`
10. https://developer.mozilla.org/en-US/docs/Web/CSS
11. https://sharifov.wordpress.com/2017/12/12/domdocument-object-model/
12. JavaScript Loops Explained: For Loop, While Loop, Do...while Loop, and More (https://www.freecodecamp.org/news/javascript-loops-explained-for-loop-for/)
13. https://zahirmirzamammadli.medium.com/javascript-functions-funksiyalar-7a93c7bf3cde
14. Web Storage API (https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API)
15. https://owasp.org/www-project-top-ten/
16. Bash Reference Manual (https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html#Quoting)
17. LDAP Injection Prevention Cheat Sheet (https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.htn
18. What is a SYN flood attack? (https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/)
19. DoS / DDoS Saldırısı Nedir? Korunma Yöntemleri Nelerdir? (https://berqnet.com/blog/dos-ddos-saldirisi-nedir)
20. https://csrc.nist.gov/glossary/term/authentication
21. https://owasp.org/www-community/attacks/SQL_Injection
22. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html