Web Security

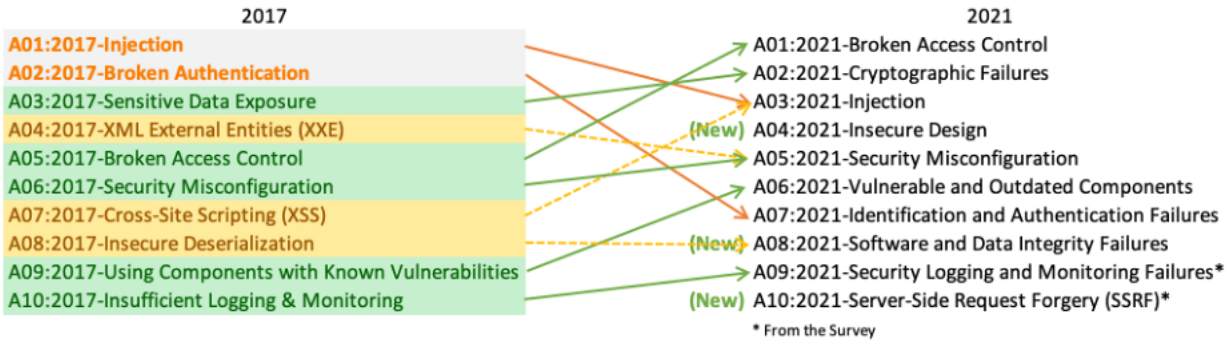
Bu gün demək olar ki, hər bir şirkətin, ölçüsündən asılı olmayaraq, xarici şəbəkə sərhədlərində bir və ya bir neçə **veb tətbiqi** mövcuddur. Bu tətbiqlər sadə **statik veb saytlar**dan, WordPress kimi **Content Management System (CMS)**-lə işləyən bloglara, eləcə də müxtəlif **istifadəçi rollarını** dəstəkləyən mürəkkəb **sign-up/login funksionallığı** olan tətbiqlərə qədər dəyişir — əsas istifadəçilərdən tutmuş super adminlərə qədər.

Hal-hazırda bir **xarici hostun** açıq şəkildə **tanınmış public exploit** vasitəsilə istismar edilə biləcəyini görmək nadirdir (məsələn, zəif xidmət və ya Windows-da **remote code execution (RCE)** zəifliyi), lakin bu baş verə bilər. **Veb tətbiqlər geniş hücum səthi təqdim edir** və onların dinamik təbiəti onları **daima dəyişən və bəzən gözdən qaçan** hala gətirir. Nisbətən sadə bir kod dəyişməsi belə **katastrofik zəifliklər** və ya bir sıra zəifliklər yarada bilər ki, bunlar birləşdirilərək **həssas məlumatlara çıxış** və ya **web serverdə və ya mühitdəki digər hostlarda (məsələn, database serverlər) uzaqdan kod icrası** əldə etməyə imkan verə bilər.

Uğurlu bir **veb tətbiq hücumu** ciddi itkilərə və böyük biznes fasilələrinə səbəb ola bilər. Çünki veb tətbiqlər **serverlərdə işləyir**, bu serverlər bəzən **həssas məlumatlar** saxlayır və çox vaxt **istifadəçi və ya şirkət məlumatlarını ehtiva edən verilənlər bazaları ilə bağlıdır**. Əgər sayt uğurla hücuma məruz qalarsa, bütün bu məlumatlar təhlükəyə düşə bilər. Buna görə də hər bir biznes üçün **veb tətbiqləri düzgün şəkildə test etmək, zəiflikləri tapmaq və onları vaxtında düzəltmək** kritik əhəmiyyət daşıyır. Eyni zamanda düzəlişin mövcud zəifliyi həll etdiyini və yeni problemlər yaratmadığını test etmək lazımdır.

OWASP TOP 10

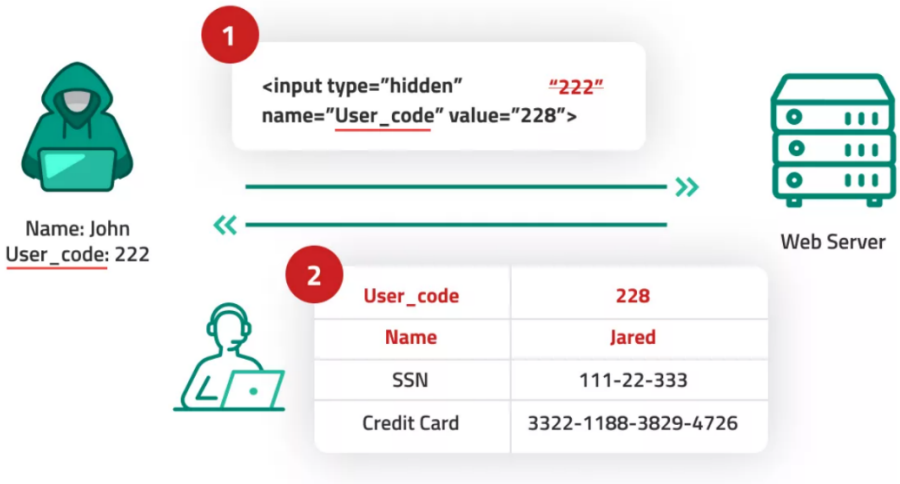
Əgər **İT (İnformasiya Texnologiyaları)** sahəsində işləyiriksə, tətbiqlərimizin **nə qədər təhlükəsiz olduğunu** bilməliyik. Tətbiqlərin təhlükəsizliyi deyilən zaman ağıla ilk OWASP **(**Open Web Application Security Project) TOP 10 hal hazırda çox yayılmış 10 veb proqram zəifliyinin siyahısı gəlir. Bu siyahı sayəsində istifadəçilər ən kritik risklər və təhdidlər, onların nəticələri və əks tədbirlərdən mılumatlı olurlar. Bu siyahı proqramçılar və təhlükəsizlik mütəxəssisləri üçün çox vacib bir yol xəritəsidir. **OWASP** əslində bir **təşkilatdır** və **10 ildən artıqdır ki, veb təhlükəsizliyi** sahəsində **resurslar, konfranslar və təlimlər** kimi xidmətlər təqdim edir. OWASP siyahısı hər üç-dörd ildən bir yenilənir. 2021-ci ilin ən yaxşı 10 OWASP zəifliyi aşağıdakı şəkildə göstərilmişdir.



1. (Broken Access Control)

Bu zəiflikdə, tətbiq istifadəçilərin nəyə daxil ola biləcəyini və nəyi edə biləcəyini düzgün şəkildə yoxlamır. Nəticədə, adi bir istifadəçi idarəçi (administrator) funksiyalarına və ya başqa bir istifadəçinin məlumatlarına daxil ola bilir.

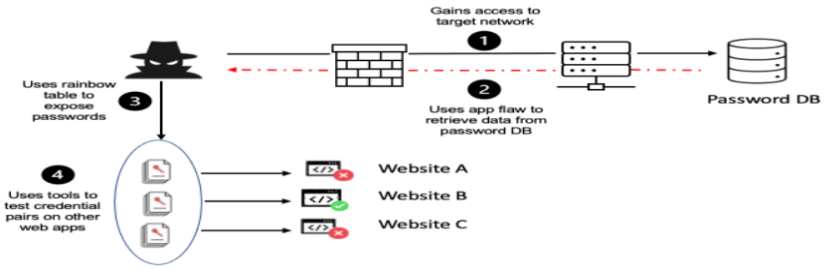
Nümunə: Tutaq ki, bir onlayn mağazada sifarişlərinizi yoxlamaq üçün /user/myorders?orderId=123 ünvanına daxil olursunuz. Zəif təhlükəsizlik olan bir saytda, bu URL-dəki 123 rəqəmini 124 ilə əvəz etməklə başqa bir istifadəçinin sifarişinə baxa bilərsiniz. Yaxşı təhlükəsizlikdə isə, sistem 124 nömrəli sifarişin sizə aid olmadığını yoxlayıb, girişi əngəlləməlidir.



2. Kriptoqrafik Uğursuzluqlar (Cryptographic Failures)

Həssas məlumatların (şifrələr, kredit kartları, şəxsi məlumatlar) düzgün şəkildə şifrlənmədiyi və ya şifrələrin zəif şifrləmə alqoritmləri ilə qorunduğu zaman yaranan problemdir. Hücumçular bu zəiflikdən istifadə edərək bu məlumatları əldə edə bilərlər.

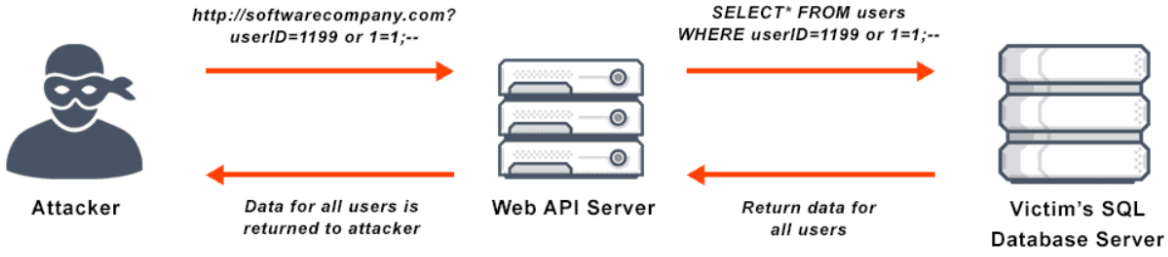
Nümunə: Bir veb sayt istifadəçilərin şifrələrini sadə, yəni şifrələnməmiş (plaintext) formatda məlumat bazasında saxlayır. Bu məlumat bazası hər hansı bir şəkildə ələ keçirilərsə, bütün istifadəçilərin şifrələri açıq şəkildə görünər. Düzgün yanaşma, şifrələrin güclü hash alqoritmləri (məsələn, bcrypt) istifadə edərək saxlanılmasıdır.



3. İnyeksiya (Injection)

Bu, ən yaygın və təhlükəli zəifliklərdən biridir. Hücumçu məlumat giriş sahələrinə (məsələn, axtarış çubuğu və ya giriş forması) xüsusi kodlar daxil edərək, tətbiqin daxili sorğularını dəyişdirməyə çalışır. SQL İnjeksiya, LDAP İnjeksiya və ya Komanda İnjeksiya kimi növləri var.

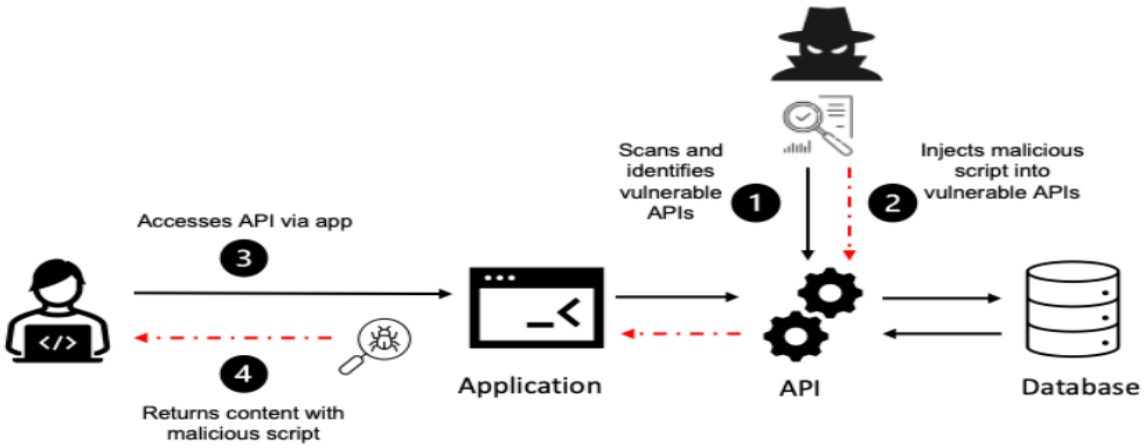
Nümunə: Bir giriş formasına istifadəçi adı olaraq ' OR '1'='1'-- yazılır. Əgər tətbiq bu girişi düzgün yoxlamırsa, SQL sorğusu belə bir formaya çevrilə bilər: SELECT * FROM users WHERE username = '' OR '1'='1'-- AND password = ''. Bu sorğu həmişə doğru olduğu üçün, hücumçu istifadəçi adı və şifrə bilmədən sistemə daxil ola bilər.



4. Təhlükəsiz olmayan Dizayn (Insecure Design)

Bu, kod yazılarkən yaranan xətadan daha çox, tətbiqin təhlükəsizlik prinsipləri nəzərə alınmadan dizayn edilməsidir. Təhlükəsizlik yoxlamalarının planlaşdırılmaması və ya kritik funksiyaların sadəcə "etibar edilən" yoxlamalara buraxılması bu zəifliyin səbəbidir.

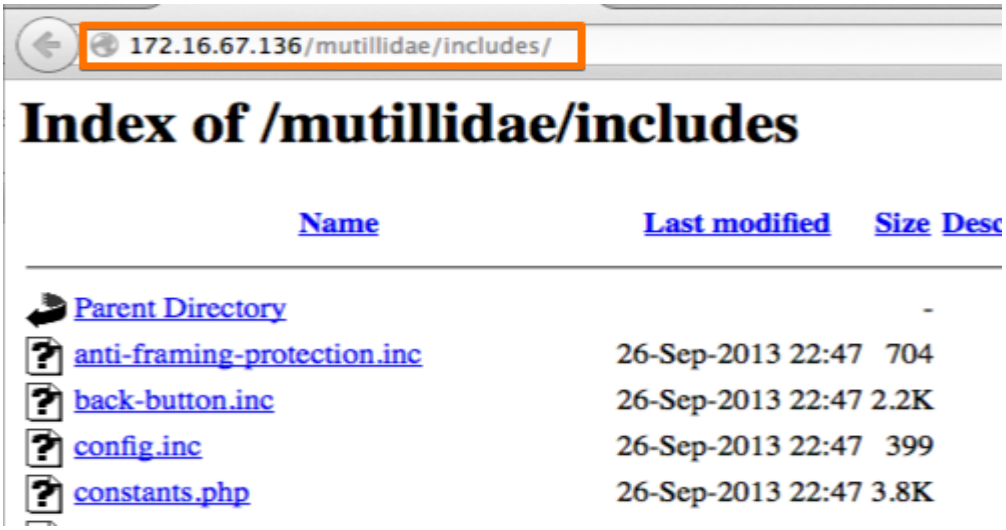
Nümunə: Bir proqramçının kredit kartı məlumatlarını bir neçə fərqli xidmətə ötürən bir modulu yazması. Ancaq bu moduldakı məlumatların şifrlənməsi və ya ötürülməsi üçün təhlükəsizlik prinsipləri nəzərə alınmır. Modul işləsə də, əslində təhlükəli bir şəkildə dizayn edilib.



5. Yanlış Konfiqurasiya (Security Misconfiguration)

Server, veb tətbiqi və ya digər komponentlərin düzgün qurulmaması nəticəsində yaranır. Bu, standart şifrələrin dəyişdirilməməsi, gərəksiz xidmətlərin aktiv qalması və ya səhv mesajlarının hücumçuya çox məlumat verməsi kimi halları əhatə edir.

Nümunə: Bir veb serverin quraşdırılması zamanı **debug (hata ayıklama)** rejimi aktiv buraxılır. Bu zaman hər hansı bir xəta baş verdikdə, server səhvin bütün detallarını (fayl yolları, məlumat bazası sorğuları və s.) ekrana yazır. Hücumçular bu məlumatlardan istifadə edərək sistem haqqında əlavə məlumat əldə edə bilərlər.



6. Köhnəlmiş və ya Təhlükəsiz olmayan Komponentlər (Vulnerable and Outdated Components)

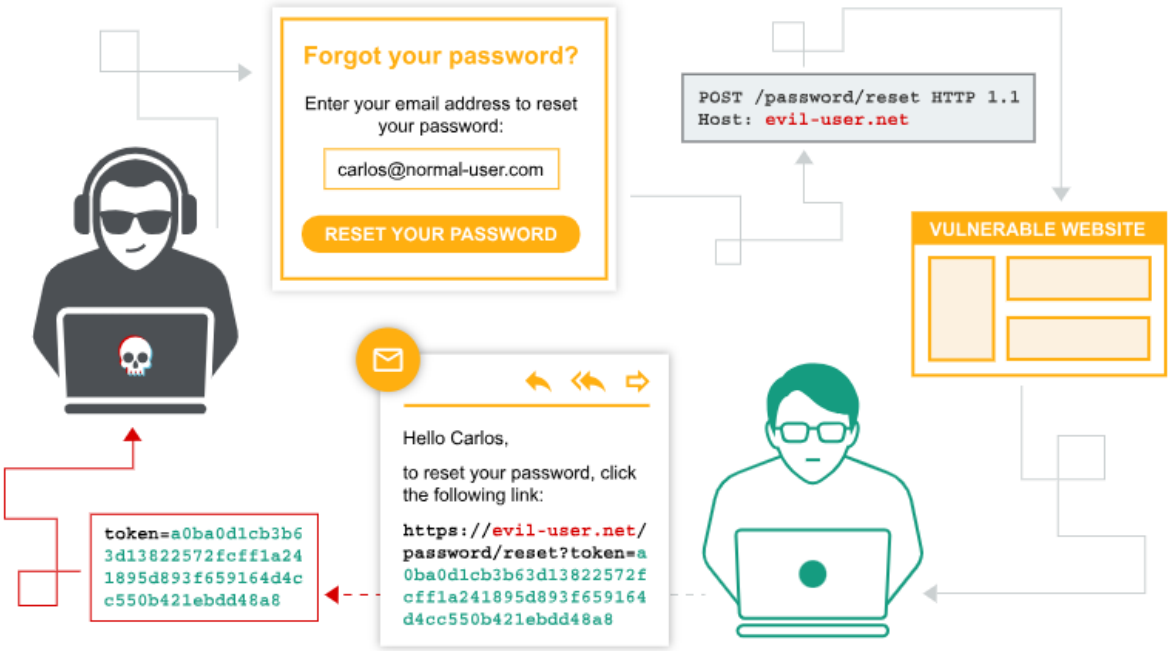
Tətbiqə daxil edilən üçüncü tərəf kitabxanalar, freymvorklar və ya modulların köhnəlmiş versiyalarıdır. Bu köhnəlmiş komponentlərin özlərində təhlükəsizlik zəiflikləri ola bilər.

Nümunə: Tətbiq bir illər əvvəl yayımlanmış və bir çox zəifliyi olan bir JavaScript kitabxanasının köhnə versiyasını istifadə edir. Hücumçu bu zəiflikləri bilərək, tətbiqə zərər verə bilər. Bu halda problem proqramçının öz kodunda deyil, istifadə etdiyi köməkçi komponentdədir.

7. İdentifikasiya və Kimlik Doğrulamada Uğursuzluqlar (Failures)

Tətbiqin istifadəçini düzgün şəkildə müəyyənləşdirməsində (identifikasiya) və ya istifadəçinin kimliyini təsdiqləməsində (kimlik doğrulaması) yaranan problemlərdir. Bu, zəif şifrə siyasətləri, şifrə bərpası mexanizmlərindəki boşluqlar və ya sessiya idarəçiliyindəki səhvlər ola bilər.

Nümunə: Bir veb sayt istifadəçinin şifrə bərpa etməsi üçün cavab verməli olduğu gizli sualın cavabını brauzerə avtomatik olaraq yazdırır. Bu halda hücumçu istifadəçinin adını bilərək gizli sualın cavabını brauzerin yaddaşından tapa bilər.



8. Proqramlaşdırma və Məlumat Bütövlüyü Pozulmaları (Software and Data Integrity Failures)

Proqram kodunun və ya məlumatların düzgün şəkildə yoxlanılmaması nəticəsində yaranan problemlər. Bu, təhlükəsiz olmayan avtomatik yenilənmələr və ya etibarsız mənbələrdən gələn məlumatların qəbul edilməsi kimi halları əhatə edir.

Nümunə: Bir veb tətbiqi, yenilənmələrini yoxlamaq üçün etibarsız bir HTTP bağlantısı (HTTPS əvəzinə) istifadə edir. Hücumçu, bu ötürməni ələ keçirərək, tətbiqin yenilənmə faylını zərərli bir faylla əvəz edə bilər.

9. Təhlükəsiz olmayan Təşkilatlandırma (Security Logging and Monitoring Failures)

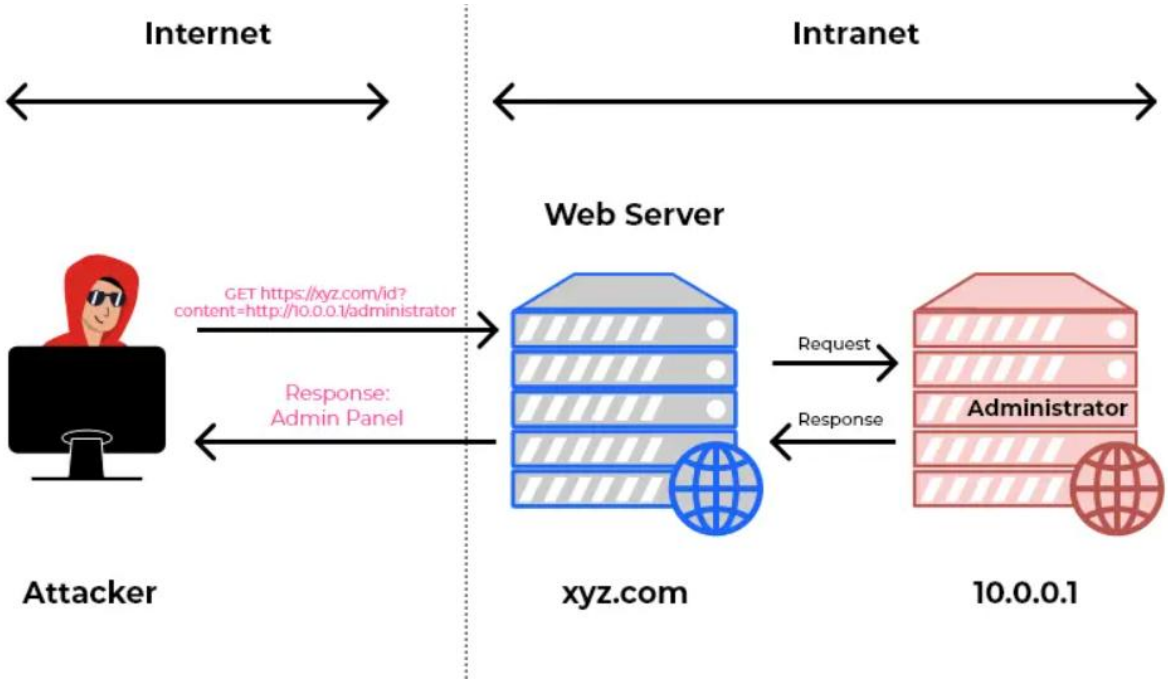
Sistemdə baş verən hadisələrin (uğurlu və ya uğursuz giriş cəhdləri, giriş səhvləri, və s.) düzgün qeydə alınmaması və ya qeydə alınmış məlumatların effektiv şəkildə izlənilməməsi. Bu səhvlər, bir hücumun aşkar edilməsini və ya araşdırılmasını çətinləşdirir.

Nümunə: Bir veb tətbiqində çox sayda uğursuz giriş cəhdi baş verir, ancaq sistem bu cəhdləri qeydə almır və ya heç bir xəbərdarlıq yaratmır. Nəticədə, kiberhücum (məsələn, Brute-Force) saatlarla davam edə bilər və sistem idarəçisi bundan xəbərsiz qalar.

10. Server tərəfindən saxta sorğu yaranması (Server-Side Request Forgery - SSRF)

Bu, serverin, hücumçunun verdiyi URL-ə əsasən daxili şəbəkədəki bir resursa sorğu göndərməyə məcbur edilməsidir. SSRF zəifliyi olan bir tətbiqdən istifadə edən hücumçu daxili şəbəkədəki gizli serverlərə və xidmətlərə giriş əldə edə bilər.

Nümunə: Bir tətbiq istifadəçilərə URL verib, həmin URL-dəki faylı yükləməyə imkan verir. Hücumçu bu funksiyadan istifadə edərək serverə http://127.0.0.1/admin kimi daxili bir URL göndərir. Server bu daxili URL-ə sorğu göndərərək hücumçuya admin panelinin məlumatlarını qaytara bilər. Normalda isə bu cür sorğular xarici şəbəkədən edilə bilməz.



Front-End Vulnerabilities

Cross-site scripting (XSS)

Simply put, XSS is one of the most important vulnerabilities out there. It's both incredibly common and extremely powerful, especially when used as part of a wider exploit chain. This is a huge topic, with plenty of labs for complete beginners and seasoned pros alike.

Go to topic → 30 Labs

Cross-site request forgery (CSRF)

Go to topic → 12 Labs

Cross-origin resource sharing (CORS)

Go to topic → 3 Labs

Clickjacking

Go to topic → 5 Labs

DOM-based vulnerabilities

Go to topic → 7 Labs

WebSockets

Go to topic → 3 Labs

Back-End Vulnerabilities

SQL injection

SQL injection is an old-but-gold vulnerability responsible for many high-profile data breaches. Although relatively simple to learn, it can potentially be used for some high-severity exploits. This makes it an ideal first topic for beginners, and essential knowledge even for more experienced users.

Go to topic → 18 Labs

Authentication

Go to topic → 14 Labs

Path traversal

Go to topic → 6 Labs

Command injection

Go to topic → 5 Labs

Business logic vulnerabilities

Go to topic → 11 Labs

Information disclosure

Go to topic → 5 Labs

Access control

Go to topic → 13 Labs

File upload vulnerabilities

Go to topic → 7 Labs

Race conditions

Go to topic → 6 Labs

Server-side request forgery (SSRF)

Go to topic → 7 Labs

XXE injection

Go to topic → 9 Labs

NoSQL injection

Go to topic → 4 Labs

API testing

Go to topic → 5 Labs

Web cache deception

Go to topic → 5 Labs