XYZ A.Ş.

Kaynak Kod Analizi Raporu

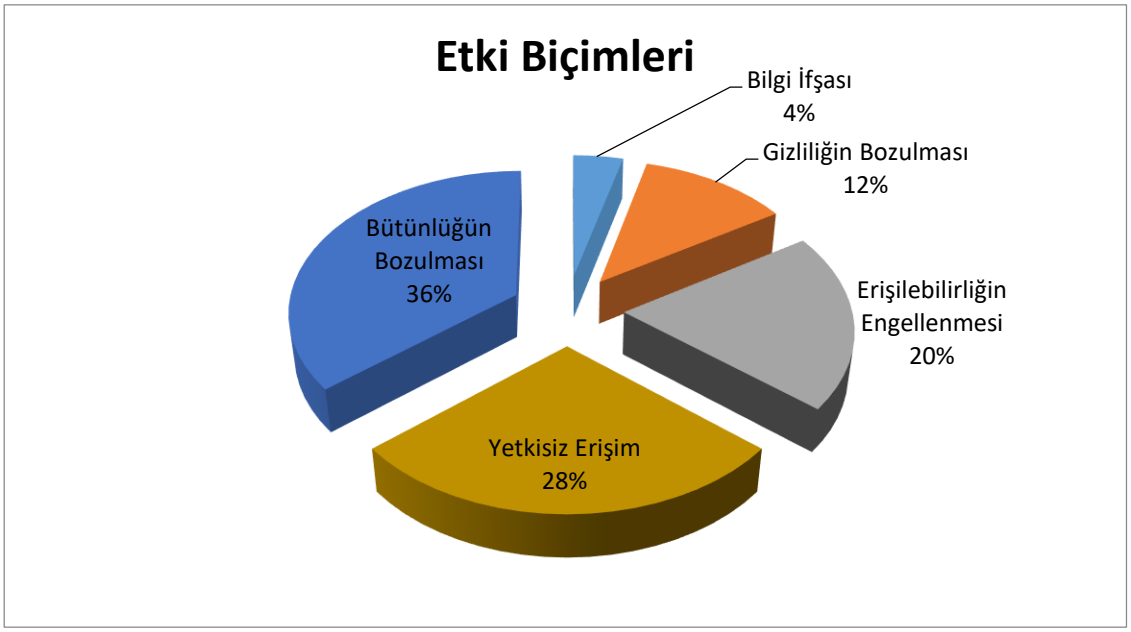
Ağustos 2019

1 Giriş

1.1 Yönetici Özeti

Kapsam içerisinde yer alan kurum bilişim altyapısını oluşturan bilgi teknolojileri bileşenlerinin güvenliklerinin denetlenmesi ile oluşturulan bu raporda bir saldırgan veya zararlı yazılım tarafından gerçekleştirilebilecek saldırılara karşı zafiyetler tespit edilmiş ve olası sonuçlar incelenmiştir.

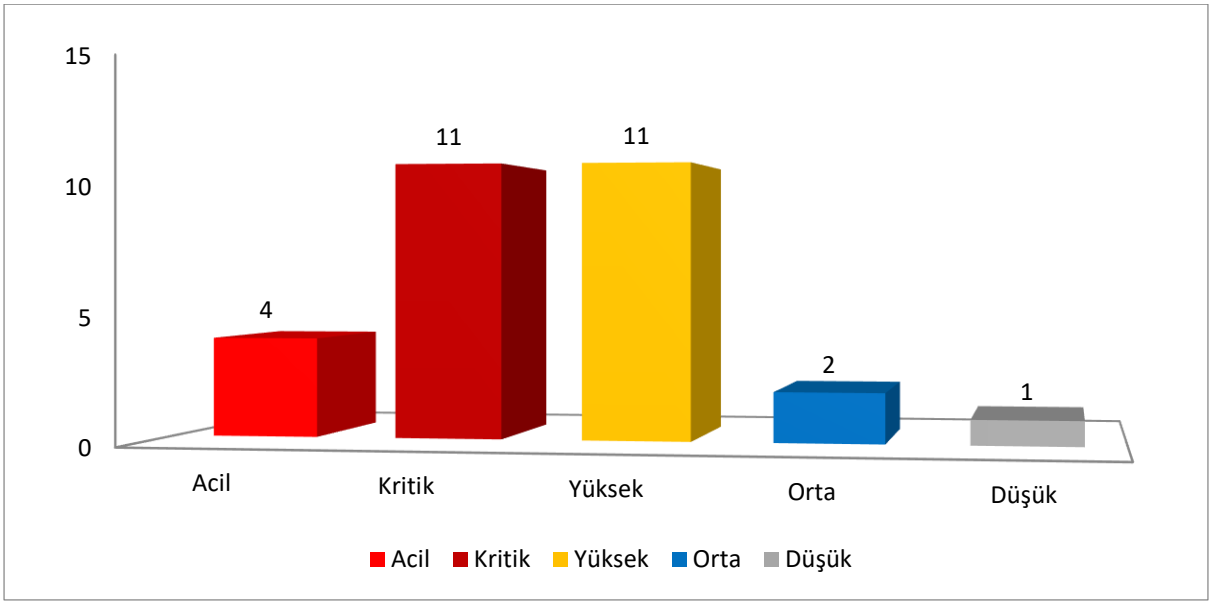
Bir saldırganın doğrudan veya dolaylı olarak servislere erişebileceği, veri çalabileceği veya manipüle edebileceği tespit edilmiştir. Keşfedilen zafiyetlerin etki biçimleri şu şekildedir; **Bilgi İfşası 1 (%4), Gizliliğin Bozulması 3 (%12), Erişilebilirliğin Engellenmesi 5 (%20), Yetkisiz Erişim 7 (%28), Bütünlüğün Bozulması 9 (%36).**



Tablo 1 : Keşfedilen Zafiyetlerin Etki Biçimi Dağılımları

Uygulama üzerinde sıkılaştırma eksikliklerinden kaynaklanan, daha genel manada tasarım ve kodlama eksiklikleri kaynaklı bulguların bulunduğu görülmüştür.

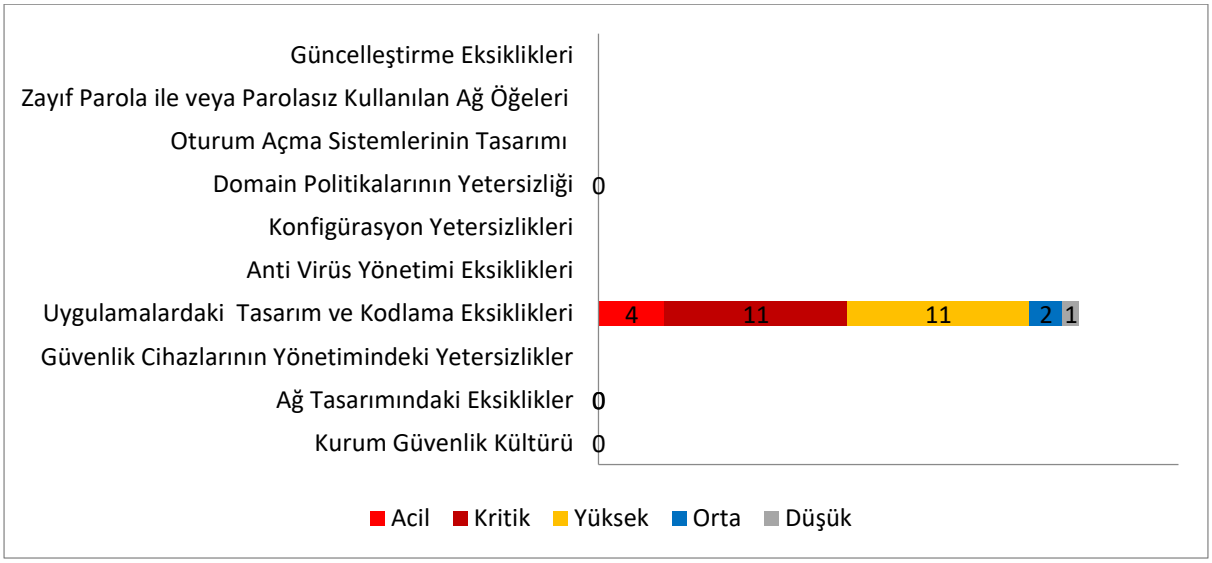
Yapılan değerlendirme sonucunda kapsamda bulunan sistemlerin bazı önemli güvenlik zafiyetlerine sahip olduğu tespit edilmiştir. Bu nedenle güvenlik açıkları yakından takip edilmeli ve mümkün olan en kısa sürede sistemlerden kaldırılmalıdır.



Tablo 2 : Bulguların Kritiklik Seviyesine Göre Dağılımları

Yapılan incelemelerde dikkat çeken bazı "Kök Nedenler" şu şekilde sıralanmıştır:

1. **Uygulamalardaki Tasarım ve Kodlama Hataları**
2. **Konfigürasyon Yetersizlikleri**



Tablo 3 : Kök Neden Dağılım Grafiği

Güvenlik denetimleri düzenli olarak devam ettirilmeli, güncel tehditler göz önüne alınarak kullanılan altyapı, yapılandırma ayarları ve uygulamalar güncelleştirilmelidir.

Sunulan kök nedenlere ait iyileştirme çalışmaları planlanmalı ve mümkünse bir Bilgi Güvenliği Yönetim Sistemi dahilinde uygulamaya konmalıdır.

Bu çalışmalar test kapsamında bulunan bilgi varlıklarının denetim tarihindeki durumunu göstermektedir.

1.2 Amaç

Uyguladığı politikalar ile müşterilerinin ihtiyaç ve beklentilerini en iyi şekilde karşılamak, kalite standartları yüksek ürün ve hizmetleri sunarak maksimum düzeyde müşteri memnuniyeti sağlamak, her ticari kurumun hedeflerinden biridir. Bu amaç doğrultusunda kurum sermayesinin bir parçası olan, müşterilere ve kuruma ait bilgilerin korunması, hem kalite hem de iş sürekliliği bakımından önem verilmesi gereken bir konudur.

Operasyonel olarak erişilebilir, gizliliği ve bütünlüğü korunuyor olması hedeflenen bilgi varlıkları ve bilgi sistemlerinin mevcut güvenlik seviyelerini belirleyerek, olası problem ve çözüm önerilerini raporlamayı amaçlayan “Kaynak Kod Analizi” çalışması, **2019 Yılı Ağustos ayında** gerçekleştirilerek bu rapor hazırlanmıştır.

1.3 Hedef

Bu çalışmanın hedefi, kapsam dahilindeki bilişim teknolojileri altyapısı ile kuruma ait bilgi varlıklarının güvenliklerinin denetlenmesi yoluyla bu bileşenleri oluşturan öğelerin güvenlik seviyesinin artırılmasını sağlayacak önerileri ortaya koymaktır.

1.4 Kaynak Kod Analizi Metodolojisi

İnternet Güvenliği kaynak kod analizinde izlenen test yönteminde, bir statik kaynak kod analiz aracı ile birlikte, kodun ağaç yapısı içerisinde ki akışının belirlenmesi, üzerinde bulunabilecek zafiyetlerin ayrıştırılması, kodun kalitesinin değerlendirilmesi ve kaynak yönetimini içeren bulgu maddelerinin otomatik olarak belirlenmesi amaçlamaktadır. Bu süreç zarfında çıkan bulgular elle yapılan kontrollerle doğrulukları ispatlanarak testler tamamlanmaktadır.

Statik Tarama Aşaması

Bu aşamada gerçekleştirilen işlemler aşağıda listelenmiştir;

- Kaynak kod üzerinde bulunan konfigürasyon hatalarının tespiti,
- Bilinen güvenlik açıklarının tespit edilmesi,
- Dizin içerik listeleme problemlerinin belirlenmesi,
- Dizin atlatma problemlerinin tespiti,
- Kaynak kodun kalitesinin tespiti,
- Kaynak kod üzerinde kullanılan kaynakların yönetiminin tespiti,
- Web sunucusu hata denetim mekanizmasının kontrol edilmesi.

Doğrulama Aşaması

Doğrulama aşaması, otomatik tarama sonrasında tespit edilen güvenlik açıklarının uzmanlarımız tarafından kaynak kod üzerinde gerçekten var olup olmadığının tespit edilmesi aşamasıdır. Bu aşama hatalı sonuçların minimuma indirilmesi ve raporların en doğru sonuçları yansıtması açısından oldukça önemlidir. Bu aşamada hatalı sonuçların ayıklanması işlemi gerçekleştirilmektedir.

1.5 Tanım

Statik kod analizi, yazılımın çalıştırılmasına ihtiyaç duyulmadan yapılır ve bu sayede, uygulama çalışırken fark edilemeyen hataların tespit edilmesi sağlanır.

Sistemler içerisinde kilit noktada olan yazılımlar geliştirilirken yapıları karmaşıklaşır ve bu nedenle statik kod analizi yazılım geliştirmenin bir parçası olmalıdır. Ancak bilinmelidir ki statik kod analizi, kod içerisinde bulunan her sorunu tespit etmek yerine koddaki risk türlerini tespit eder. Unutulmamalıdır ki, analiz sürecinde tüm zafiyetlerin bulunması her zaman mümkün değildir. Eğer analiz sonucunda hiçbir sorun ortaya çıkmazsa, uygulamanın %100 olarak saldırılara karşı korunduğu sonucuna varılamaz. Güvenli kod analizi kesin bir çözüm değildir fakat uygulamayı korumak için risk azaltma programının güçlü bir parçasıdır.

Yapılan analiz sırasında, analizi gerçekleştiren personel güvenlik sorunlarını tespit etmek için uygulama kodunu analiz eder ve bulguları zafiyet kategorilerine göre kategorize eder. Her bulgu uygun risk derecesine (yüksek, orta, düşük, bilgilendirme) atanır.

2 Bulgular

2.1 Kod Enjeksiyonu

Etkisi	Yetkisiz Erişim

Bulgu Açıklaması

Uzaktan kod çalıştırma saldırıları, uygulamaların arkaplanında işlem yapan servislere karşı yapılan bir saldırı türüdür. Kullanıcıdan alınan girdi değerleri zararlı karakter olarak adlandırılan **$()+=;** gibi karakterlerden arındırılmadan direkt olarak çalıştırıldığında beklenmeyen değerlerin kod üzerinde çalıştırma yetkisi kazanması, tüm uygulamanın işleyişinin saldırganın istediği yöne döndürülmesiyle sonuçlanır.

Bu tür saldırılarda amaç her zaman, uygulamanın yazıldığı yazılım dilini kullanarak işletim sistemi üzerinde kod çalıştırıp, uygulamanın barındığı sunucuda tam yetkiye sahip olarak her türlü bilgiyi ele geçirmek ve kalıcılık elde etmekdir ve en tehlikeli saldırı vektörü olarak sınıflandırılır.

Çözüm Önerileri

Code injection saldırılarından korunmak için;
• Veri tipinin yanı sıra yollanan veri büyüklüğünün de izin verilen aralıkta olup olmadığı kontrol edilmelidir. Bu kontrol özellikle gelen veri uygulama tarafından işlenmeden önce veri tipine göre mutlaka kontrol edilmelidir. Eğer olması gereken haricinde bir veri tipine sahip veri yollanmış ise istek kabul edilmemelidir. Örneğin sayısal bir değer girilen yere alfa numerik karakterler yollanmış ise bu istek kabul edilmemelidir. Veri tipi kontrolü mutlaka yapılmalıdır.
• Uygulamaya yollanabilecek karakterlerin ne olduğu tanımlanmalı, bunlar dışındaki her karakter işlenmeden önce veri içinden temizlenmeli veya istek komple reddedilmelidir. İzin verilmeyen karakterlerin tanımlanması yerine sadece izin verilen karakterlerin belirlenmesi daha başarılı bir koruma sağlayacaktır.

CVE/CWE	CWE-96
Ek Bilgi	**Code Injection (OWASP)** https://www.owasp.org/index.php/Code_Injection **Code Injection (Wikipedia)** https://en.wikipedia.org/wiki/Code_injection **Injection Prevention Cheat Sheet (OWASP)** https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

2.2 LDAP Enjeksiyonu

Etkisi	Yetkisiz Erişim

Bulgu Açıklaması

LDAP injection saldırıları, saldırganlar LDAP servisine giden direkt sorgulara beklenmeyen girdiler üretildiğinde
gerçekleşir. Kullanıcılardan alınan girdi değerleri zararlı karakter olarak sınıflandırılan '()=+, gibi karakterlerden
arındırılmadan LDAP sorgularında işleme alındığında yetkisiz bir şekilde sorguya müdahale eden saldırganlar, LDAP servisinde uygulama ile aynı haklara sahip olarak kod çalıştırma hakkına erişirler. Bu tür ataklar 'Kritik' olarak
sınıflandırılır ve en çok hasar veren saldırılar olarak değerlendirilir.

LDAP sorgularına geçersiz girdi enjekte edildiğinde genellikle anlamlı bir hata mesajı dönmez. LDAP injection açıklığının varlığına işaret olabilecek belirti bir arama fonksiyonunun hatalı girdi sonrası HTTP 500 hata kodu gibi bir sunucu hatası döndürmesidir.

Çözüm Önerileri

LDAP injection saldırılarından korunmak için;
• Veri tipinin yanı sıra yollanan veri büyüklüğünün de izin verilen aralıkta olup olmadığı kontrol edilmelidir. Bu kontrol özellikle gelen veri uygulama tarafından işlenmeden önce veri tipine göre mutlaka kontrol edilmelidir. Eğer olması gereken haricinde bir veri tipine sahip veri yollanmış ise istek kabul edilmemelidir. Örneğin sayısal bir değer girilen yere alfa numerik karakterler yollanmış ise bu istek kabul edilmemelidir. Veri tipi kontrolü mutlaka yapılmalıdır.

• Uygulamaya yollanabilecek karakterlerin ne olduğu tanımlanmalı, bunlar dışındaki her karakter işlenmeden önce veri içinden temizlenmeli veya istek komple reddedilmelidir. İzin verilmeyen karakterlerin tanımlanması yerine sadece izin verilen karakterlerin belirlenmesi daha başarılı bir koruma sağlayacaktır.

CVE/CWE	CWE-90
Ek Bilgi	**LDAP injection Prevention CheatSheet** https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html **How to prevent LDAP injection** https://affinity-it-security.com/how-to-prevent-ldap-injection/

2.3 SQL Enjeksiyonu

Etkisi	Yetkisiz Erişim

Bulgu Açıklaması

SQL injection saldırıları, saldırganlar veritabanına giden direkt sorgulara beklenmeyen girdiler üretildiğinde gerçekleşir. Kullanıcılardan alınan girdi değerleri zararlı karakter olarak sınıflandırılan **'()+&** gibi karakterlerden arındırılmadan SQL sorgularında işleme alındığında yetkisiz bir şekilde sorguya müdahale eden saldırganlar, veritabanında uygulama ile aynı haklara sahip olarak kod çalıştırma hakkına erişirler. Bu tür ataklar 'Kritik' olarak sınıflandırılır ve en çok hasar veren saldırılar olarak değerlendirilir.

Çözüm Önerileri

SQL saldırılarından korunmak için;
• Veri tipinin yanı sıra yollanan veri büyüklüğünün de izin verilen aralıkta olup olmadığı kontrol edilmelidir. Bu kontrol özellikle gelen veri uygulama tarafından işlenmeden önce veri tipine göre mutlaka kontrol edilmelidir. Eğer olması gereken haricinde bir veri tipine sahip veri yollanmış ise istek kabul edilmemelidir. Örneğin sayısal bir değer girilen yere alfa numerik karakterler yollanmış ise bu istek kabul edilmemelidir. Veri tipi kontrolü mutlaka yapılmalıdır.

• Uygulamaya yollanabilecek karakterlerin ne olduğu tanımlanmalı, bunlar dışındaki her karakter işlenmeden önce veri içinden temizlenmeli veya istek komple reddedilmelidir. İzin verilmeyen karakterlerin tanımlanması yerine sadece izin verilen karakterlerin belirlenmesi daha başarılı bir koruma sağlayacaktır.
Benzer şekilde veritabanı kullanıcısına ait hakların kısıtlanması işletim sistemi üzerinde gerçekleştirilebilecek aktiviteleri sınırlayacağı için, sistemin komple ele geçirilmesini büyük oranda engelleyecektir.

CVE/CWE	CWE-89
Ek Bilgi	Blind_SQLInjection.pdf http://www.owasp.org/index.php/SQL_Injection SQL Injection Prevention Cheat Sheet https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

2.4 Siteler Arası Kod Çalıştırma (XSS)

Etkisi	İstemci Tabanlı Zafiyet

Bulgu Açıklaması

Dinamik olarak oluşturulan web sayfalarında, kullanıcı tarafından sağlanan girdilerin uygulama tarafından düzgün bir şekilde işlenemediği veya kontrol edilemediği durumlarda, yetkisiz kişiler istedikleri script kodlarının, görüntülenecek sayfada çalışmasını sağlayabilirler. Söz konusu problem Cross Site Scripting (XSS) olarak adlandırılmaktadır.
Bu problem ile daha çok aşağıdaki durumlarda karşılaşılır:

• Girilen arama metnini tekrar tarayıcıda görüntüleyen arama motorları,
• Hataya yol açan metni tekrar eden hata mesajları,
• Girilen değerlerin sonradan kullanıcıya görüntülendiği formlar,
• Kullanıcıların kendi mesajlarını yayınlayabilmelerine izin veren sayfalar
XSS açığı ile saldırgan, HTML kodlarının arasına istemci tabanlı kod ekleyerek, kullanıcının tarayıcısında istediği istemci tabanlı kodu çalıştırabilir. Bu açık sayesinde saldırgan, açığın bulunduğu domain ve sayfanın bilgilerini, oturum bilgilerini ve diğer obje değerlerini çalabilir. Örneğin, bir kullanıcının oturum bilgilerini çalan saldırgan o kullanıcının yetkisi dâhilindeki her işlemi yapabilir.

Çözüm Önerileri

Bu saldırıları önlemek için birkaç yöntem mevcuttur. Bu yöntemler kullanıcı girdilerinin (URL, form vs.) filtrelenmesi esasına dayanmaktadır. Kullanıcının girdilerindeki zararlı karakterler filtrelenmelidir. Bu yöntem aynı zamanda SQL enjeksiyonu, uzaktan komut çalıştırma, bellek taşması gibi diğer ataklarını da engeller. Aşağıda XSS için alınabilecek önlemlere değinilmiştir:

• Değişik parametreler aracılığı (cookie, formlar, http başlığı, kod içindeki hidden alanları, sorgu parametreleri) ile GET veya POST metodu ile yollanan veriler içinde nelere izin verileceği mutlaka belirlenmelidir.
• Oluşturulan çıktı mutlaka kontrol edilmeli ve uygun şekilde filtrelenmelidir.
• XSS için kullanılabilecek <,>,/,”,(gibi tüm özel karakterler istek içinden silinmelidir. İstek içinde yollanacak veri sadece alfanumerik karakterler olacak şekilde düzenlendikten sonra uygulamaya yollanmalıdır.
• Üretilen çıktıda özel karakterlerin silinmesi için değişik uygulamalar tarafından geliştirilmiş fonksiyonlar kullanılabilir. Örneğin PHP tabanlı uygulamalarda htmlspecialchars ASP.NET tabanlı uygulamalarda ise HTMLEncode fonksiyonları bu amaçla kullanılabilir.
PHP: string htmlspecialchars (string string [, int quote_style])
ASP / ASP.NET: Server.HTMLEncode (strHTML String)

Çok sayıda uygulamada benzer problemlerin olması, kodların kısa bir süre içinde güvenlik gereksinimlerini karşılayacak şekilde değiştirilmesini oldukça güçleştirebilir.

Bu sebeple web uygulamasına yönelik güvenlik yazılımları kullanmak, kodlar değiştirilmeden de belirtilen açıklara karşı önlem almanıza yardımcı olacaktır. Buna ek olarak sunucu hata mesajlarının standart hale getirilmesi de söz konusu açıktan doğacak riski minimuma indirecektir.

Aşağıdaki karakterlerin filtrelenmesi tavsiye edilmektedir:

[1] | (pipe)
[2] & (ampersand)
[3] ; (noktalı virgül)

[4] $ (dolar)
[5] % (yüzde)
[6] @ (at)
[7]' (tek tırnak)
[8] " (çift tırnak)
[9] \' (backsslash-tek tırnak)
[10] \" (backslash-çift tırnak)
[11] <> (üçgen parantezler)
[12] () (parantezler)
[13] + (artı)
[14] CR (Carriage return, ASCII kodu: 0x0d)
[15] LF (Line feed, ASCII kodu: 0x0a)
[16] , (virgül)
[17] \ (backslash)

CVE/CWE	CWE-79 CWE-80
Ek Bilgi	Reviewing code for XSS issues **https://www.owasp.org/index.php/Reviewing_code_for_XSS_issues** Input Validation Vulnerability **https://www.owasp.org/index.php/Testing_for_Input_Validation** Cross-Site Tracing - Protecting Businesses from a Simple Attack **https://www.sans.org/reading-room/whitepapers/malicious/cross-site-tracing-protecting-businesses-simple-attack-1140** Hummingbird CyberDOCS Cross-Site Scripting Vulnerabilities **http://www.securityfocus.com/bid/8815** Cross Site Scripting Faq **http://www.cgisecurity.com/xss-faq.html** iMPERVA Cross Site Scripting **https://www.imperva.com/resources/glossary?term=cross_site_scripting**

2.5 Statik(Hardcoded) Parola Kullanımı

Etkisi	Hassas bilgi ifşası

Bulgu Açıklaması

Yapılan incelemelerde Program kodlarının içine gömülmüş, çalışma akışı içinde değiştirilemeyecek parolalar saptanmıştır. Statik parola kullanımı kaba kuvvet saldırılarında saldırganların işine yarayacağı gibi, derlenmiş uygulamayı eline geçiren saldırgan için de tersine mühendislik yöntemi ile uygulama içerisinden çıkartmak hayli kolay olacaktır. Özellikle derlenmiş .NET uygulamaları neredeyse kaynak koda birebir olarak terse çevrilebildiğinden, gömülmüş parolalar büyük risk teşkil etmektedir.

Çözüm Önerileri

Bu tür saldırılardan korunmak için;

Belirli bir algoritmaya göre oluşturulan ve belirli zaman aralıklarında geçerli olan parolalar üretilip, şifrelenmiş bağlantılar aracılığı ile sunucuya gönderilmelidir.
Üretilen kullanıcıların yetkileri sunucu tarafında sadece yapmaları gereken iş kadar kısıtlandırılmalıdır.

CVE/CWE	CWE-798
Ek Bilgi	**Use of Hardcoded Passwords (OWASP)** https://www.owasp.org/index.php/Use_of_hard-coded_password **Hardcoded Credentials** https://www.veracode.com/blog/managing-appsec/hardcoded-credentials-why-so-hard-prevent **Use of Embedded Passwords** https://www.beyondtrust.com/resources/glossary/hardcoded-embedded-passwords

2.6 Path Traversal

Etkisi	Hassas bilgi ifşası

Bulgu Açıklaması

Dizin geçişi (veya yol geçişi), kullanıcı tarafından sağlanan girdilerin yetersiz sanitize edilmesinden kaynaklanır.

Bu saldırının amacı, dosya sistemine yetkisiz erişim sağlamak için zafiyetten etkilenen bir uygulamayı kullanmaktır. Bu saldırı, koddaki bir hatayı kötüye kullanmak yerine, bir güvenlik eksikliğinden (yazılım aslında tam olarak çalışması gerektiği gibi çalışıyordur) yararlanır.

Dizin geçişi ayrıca ../ (nokta nokta eğik çizgi) saldırısı, dizin tırmanma ve geriye doğru izleme olarak da bilinir. Bu saldırının bazı formları aynı zamanda kanonikleştirme saldırılarıdır.

Çözüm Önerileri

Bu saldırıları önlemek için birkaç yöntem mevcuttur. Bu yöntemler kullanıcı girdilerinin (URL, form vs.) filtrelenmesi esasına dayanmaktadır. Kullanıcının girdilerindeki zararlı karakterler filtrelenmelidir. Bu yöntem aynı zamanda SQL enjeksiyonu, uzaktan komut çalıştırma, bellek taşması gibi diğer ataklarını da engeller.

Aşağıdaki karakterlerin filtrelenmesi tavsiye edilmektedir:

[1] | (pipe)
[2] & (ampersand)
[3] ; (noktalı virgül)
[4] $ (dolar)
[5] % (yüzde)
[6] @ (at)
[7]' (tek tırnak)
[8] " (çift tırnak)
[9] \' (backsslash-tek tırnak)
[10] \" (backslash-çift tırnak)
[11] <> (üçgen parantezler)
[12] () (parantezler)
[13] + (artı)
[14] CR (Carriage return, ASCII kodu: 0x0d)
[15] LF (Line feed, ASCII kodu: 0x0a)
[16] , (virgül)
[17] \ (backslash)

CVE/CWE	CWE-23
Ek Bilgi	**Path Traversal (OWASP)** https://www.owasp.org/index.php/Path_Traversal **Directory Traversal (Medium)** https://medium.com/@mazlumagar/web-for-pentester-directory-traversal-b%C3%B6l%C3%BCmleri-da2825e47d40

2.7 Heap Inspection

Etkisi	Yetkisiz Erişim

Bulgu Açıklaması

Program içinde oluşturulan değişkenlerde tutulan parolalar, program açık olduğu süre boyunca RAM adreslerinden silinmez ve şifrelenmemiş(açık) bir şekilde muhafaza edilirler. Cihaza uzaktan veya fiziki olarak erişebilen bir saldırgan daha sonra bu parolaları tutulan RAM adreslerinden rahatlıkla çıkartabilir.

Çözüm Önerileri

Bu tür saldırıların önüne geçebilmek için örnek senaryomuzda:

string **parola** = userPASS;
Login(username,userPASS);

parola = randomValue; // Çözüm satırı

olarak bir **parola** değişkenimizin olduğunu varsayalım ve giriş yapmak için kullanıldığını varsayalım. Uygulamaya login işleminden sonra 'parola' değişkeninin üzerine farklı bir değer veya boş karakterler yazarak parolanın RAM adreslerinde saklanmasının önüne geçilebilir.

CVE/CWE	CWE-244
Ek Bilgi	**Is it more secure to overwrite the values in the char** https://security.stackexchange.com/questions/74718/is-it-more-secure-to-overwrite-the-value-char-in-a-string/75891#75891 **Heap inspection** https://stackoverflow.com/questions/30341327/heap-inspection-security-vulnerability **Java Heap inspeciton** https://community.microfocus.com/t5/Fortify-User-Discussions/java-Privacy-Violation-Heap-Inspection/td-p/1668411

2.8 Açık URL Yönlendirme

Etkisi	İstemci Tabanlı Zafiyet

Bulgu Açıklaması

İstemci tabanlı zafiyetler sunucuyu değil, uygulamayı kullanan kullanıcıları hedef alır.
Saldırgandan alınan veri girdisinin sistem tarafından gerekli kontroller yapılmadan yönlendirilme işleminin yapılması, saldırganın bu URL adresini kullanarak başka kullanıcıları zararlı web adreslerine yönlendirmesine olanak tanır.

Saldırıya uğrayan kullanıcı URL de sizin web adresini görmesine rağmen, tıkladığı linkte başka ve zararlı bir web adresine yönlendirilecektir.

Çözüm Önerileri

Bu URL yönlendirme zafiyetinden korunmak için ve güvenli şekilde gerçekleştirmek için çeşitli önlemler alınabilir.

Öncelikli olarak gerekmedikçe bu yönlendirme işleminden kaçınılması gerekmektedir.
Kullanılmasının gerektiği durumlarda kullanıcılardan yönlendirme işleminde kullanılacak olan URL'in girişi için herhangi bir alan bırakılmamalıdır.
Bunun olduğu durumlarda ise alınan verilerin düzgün doğrulama işlemlerinden geçirilmesine özen gösterilmelidir.
Doğrulama işlemlerine ek olarak girilen URL'in uygulama içerisinde olup olmadığı ve o kullanıcının bu yönlendirmeye yetkisi var mı kontrol edilmelidir.
Bu doğrulama işlemi için güvenli URL yönlendirmelerin bir **whitelist** ya da regex şeklinde oluşturulup girişlerden gelen değerlerin bunlar ile karşılaştırması tercih edilebilir.

Son olarak bütün yönlendirme işlemlerinden önce kullanıcılara siteden ayrıldığına dair bir sayfaya yönlendirip bu durumu kabul etmeleri durumda yönlendirme işlemi gerçekleştirmek bu tür saldırılardan gelebilecek zararı minimize edecektir.

CVE/CWE	CWE-601
Ek Bilgi	**Unvalidated Redirects and Forwards** https://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards **What is Open Redirect And How To Prevent it ?** https://dzone.com/articles/what-is-an-open-redirection-vulnerability-and-how **Understanding and Discovering Open Redirect Vulnerabilities** https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/understanding-and-discovering-open-redirect-vulnerabilities/

2.9 Hata Mesajları ile Bilgi İfşası

Etkisi	Uygulama hakkında bilgi edinme

Bulgu Açıklaması

Üretilen hata mesajlarını herhangi bir kategorizasyona sokmadan direkt olarak kullanıcı karşısına çıkartmak bazı durumlarda hassas verilerin ifşası ile sonuçlanabilir.
Bu tür durumlarda saldırganlar; arka plandaki değişken değerlerinden, parolalara veya RAM bilgilerine ulaşabilir.

Çözüm Önerileri

Bu tür saldırılardan korunmak için çözüm;

Kullanılan
try:
Catch:
Exception:

Bloklarında yakalanacak olan hata mesajlarını, kullanıcı dostu hata mesajlarına dönüştürmek olacaktır.

CVE/CWE	CWE-209
Ek Bilgi	**Information Exposure Through an Error Message** https://stackoverflow.com/questions/41329414/information-exposure-through-an-error-message-in-checkmarx **OWASP - Error Handling** https://www.owasp.org/index.php/Error_Handling