Zaafiyetler

a)Nedir, b)nasıl oluşur, c)etkileri nelerdir, d)kapatılması için öneriler

- *Default Credentials:*
    a. Default credentials, kurulan serviceler'de default olarak koyulmuş ve yaygın bir şekilde bilinen username:password anlamına gelir.
    b. Bir service kurulduğu zaman default credential'ları değiştirmek yerine kullanmaya devam edildiğinde oluşur
    c. saldırganlar default credential'lar ile çok kolay bir şekilde service'e erişebilir ve service'I kullanarak yada service içerisinde critic zaafiyetler bulup bunları exploit ederek, sisteme oldukça ciddi zararlar verebilirler
    d. Ne zaman bir service kurulsa, default credentials, saldırganlar tarafından tahmin edilemeyecek credential'lar ile değiştirilmelidir.

- *SQL Injection*
    a. Saldırganların, yetkisiz şekilde sistem database'inde kod çalıştırması
    b. Developer'ın user input'unu direk olarak bir sql query string'inde kullanması sonucunda oluşur
    c. Bu açık ile saldırgan sql database'indeki tüm verileri çalarak data breach'ine yol açabilir. Zaman zaman bazı durumlarda rce'ye de sebep olduğu bilinmektedir
    d. Sql query'leri framework'deki parameterized query formatında yazılmalı, bazen bu durum yeterli olmayıp developer'ın kullanıcı input'unu filtrelemesi gerekir. Mümkünse en iyisi allow list oluşturup belli başlı ve sqli'ye sebep olmayacak input'lar almak olur.

- *File Upload*
    a. Saldırganın, sistem tarafından çalıştırılan bir dil ile yazılmış kod içeren file'ı sisteme yükleyip, sonrasında o file'I ziyaret ederek kodun sistem tarafından çalıştırılması
    b. Developer'ın user tarafından yapılan file upload işleminde kontrol yapmadan file'I sisteme yüklemesi durumunda ortaya çıkar
    c. Bir saldırgan sistem'in çalıştırdığı bir file yükleyip sonrasında onu ziyaret etmesi halinde kolaylıkla rce'ye ulaşabilir. RCE ile sistemde komut çalıştırabileceği için oldukça ciddi zararlara, data breach'ine sebep olabilir
    d. User'dan alınan file'ın allow list kullanılarak, server'da kod çalıştırması mümkün olmayan file'lar kabul edilmeli

- *SSTI(Server Side Template Injection)*
    a. Saldırganın, server'ın kullandığı template içerisinde, template engine fonksiyonlarını kullanabilmesi
    b. Developer, user input'unu dynamic olarak template içinde kullandığında yada template de safe mod(input'un template fonksiyonu olarak algınamasında sorun yok modu) aktif edildiğinde ortaya çıkar
    c. Ssti kullanılarak çoğu zaman rce'ye gidilebilir
    d. User input'u template içerisinde static halde kullanılmalı ve safe mod kapalı olmalı
- *LFI*
    a. Saldırgan'ın sistemdeki file'ların içeriğini görüntüleyebilmesine olanak sağlayan bir zaafiyettir
    b. File içeriğinin gösterilmesi için yazılan bir kod'da, user input'unun direk olarak file ismini belirleyebildiği zaman ortaya çıkar
    c. Sistemdeki file'ların içeriğine ulaşılabilir
    d. Allow list kullanılarak, görüntülenebilcek file'lar kontrol edilmeli
- *RFI*
    a. Saldırganın, server'a istediği external bir file içeriğini göndermesi
    b. External bir kaynaktan, file içeriğinin gösterilmesi için yazılan bir kod'da user input'u direk olarak file url'I olarak kullanılıdığında ortaya çıkar.
    c. Sistem'de çalıştırılan kod içeren bir file yüklenerek, rce'ye gidilebilir
    d. Allow list ile external file'lar sınırlandırılmalıdır
- *Log Poisoning*
    a. Saldırganın, server log'larına malicous kod pass etmesi ve log'lara ulaşması yada başka birisinin log'u açması
    b. User log'ları bir server file'a kaydedildiği ve biri tarafından bu file'ın açılmasıyla ortaya çıkar
    c. Eğer log kayıtları .php tarzında server'ın çalıştırdığı bir dil extension'ına sahipse, saldırgan rce'ye ulaşabilir, eğer log kaydında js çalıştırılabiliyorsa, saldırgan stored xss'e sebep olabilir
    d. User input'u filtrelenmeli, output encoding, plain text log formatı yerine json formatında log'ların tutulması
- *XSS*
    a. Saldırganın, web sitesin user'larının browser'ında javascript kodu çalıştırabilmesidir

b. Developer, user input'unu direk olarak html,js,css içinde kullandığında oluşur

c. Saldırgan, diğer websitesi user'larının cookie'sini çalabilir, onlara farkında olmadıkları request'ler yaptırtabilir

d. User input'u html,js,css içinde kullanılacağı zaman output sırasında encode edilmesi gerekir

- *CSRF(Cross Site Request Forgery)*
  a. Saldırgan'ın; web sitesinin user'larına, istemedikleri bir request yaptırmasıdır

  b. Samesite=none olarak set edildiği ve developer'ın csrf-token şeklinde saldırganın tahmin edemeyeceği bir parametre kullanmadan user'dan request'I kabul ettiği zaman oluşur

  c. Saldırgan; account deletion, email changing, money transfering tarzında önemli fonksiyonlar için request'I birebir hazırlayıp kullanıcın istemedi ve farkında bile olmadığı çok ciddi request'ler yaptırarak ciddi zararlara sebep olabilir

  d. Mümkünse session cookie'si için samesite=strict yada samesite=lax olarak set edilmesi yada csrf token tarzında, saldırganın tahmin edemeyeceği random değerler önemli olan her request için gerekli olmalı

- *Command injection*
  a. Saldırgan'ın, alınan user input'u ile doğrudan webserver'ındaki shell'de komut çalıştırabilmesidir

  b. Developer aldığı user input'unu direk olarak shell komutu içinde kullandığında oluşur

  c. Saldırgan reverse shell alıp, sistemi ele geçirebilir

  d. Mümkünse user input'unu hiçbir şekilde shell komutunda kullanmamak, eğer gerekliyse allow list ile kullanı input'unu kısıtlamak

- *IDOR(Insecure Direct Object Reference)*
  a. Saldırgan kendisine ait olmayan, fonksiyon yada objelere, id değerini değiştirerek ulaşabilmesi

  b. Developer, bir fonksiyonda yada obje döneceği zaman direk olarak user input'undaki id değerini kullandığında ortaya çıkar

  c. Saldırgan yetkisiz şekilde, yapamaması gereken fonksiyonlarla diğer user'lara yada sisteme zarar verebilir, diğer user bilgilerine yetkisiz şekilde erişebilir

d. Developer user'dan aldığı id input'unun, user'a ait bir obje mi, user'ın yapmaya yetkisi olan bir fonksiyon mu şeklinde bir kontrol yapması gerekir, eğer mümkünse id mapping kullanılarak, user'dan id hiç almamak en iyi yoldur ama tabikide bu genelde sadece profil işlemleri gibi user için tamamen private durumlarda gerçekleşebilir.

- *XXE (External Entity Injection)*
  a. Saldırganın; xml data'sına external entity injecte etmesidir
  b. Developer, xml parser'a doğrudan user input'unu koyduğunda ortaya çıkar
  c. Local'deki file'ları okuma, rce gibi etkileri olaiblir
  d. Xml parser'daki external entity işleme olayını disable ederek
- *Broken Authentication*
  a. Saldırganın, authentication işlemini bypass etmesidir
  b. Bir çok farklı sebepten kaynaklanabilir, bazıları; zayıf password policies, brute-force yapılabiliyor olması, default credential kullanılması, password reset kısmındaki logic flaws, …
  c. Saldırganın; account takeover, diğer user'ların account'una yetkisiz erişebilmesi gibi ciddi sonuçlara sebep olabilir
  d. Güçlü password policy kullanımı, brute-force'un önlenmesi, password'lerin güçlü bir hash ile hash'lenerek db'de tutulması, …

- *CVE-2013-5696*
  a. Saldırganların; glpi installation sonrası, install.php'yi edit'leyebilmesi
  b. Glpi'nin 0.84.1 yada daha düşük bir versiyonu kullanıldığında ortaya çıkar
  c. Saldırgan install.php'ye code injecte ederek, sql injection ve current user 'ın prvilige'I yeterliyse php code execution gerçekleştirebilir
  d. Glpi'nin 0.84.1 sürümünden sonraki sürümlerini kullanarak sorun çözülebilir
- *CVE-2022–0847*
  a. Linux'te unprivileged bir user'ın read-only file'larına overwrite yaparak privilege escalation yapabilmesi
  b. Linux kernel, splice() system call kullanıldığında pipe ile read-only file'lara overwrite yapmamıza izin verdiği için ortaya çıkar
  c. Privilege escalation ile sistemde root olunabilir
  d. Linux kernel'i update ederek bu açığı önleyebiliriz

- *CVE-2019-16278*
    a. Saldırganın Nostromo web server'ında, directory traversal yapabilmesi
    b. User input'unun direk olarak kabul edilmesinden kaynaklı
    c. /.%0d./.%0d./.%0d./.%0d./bin/sh path'ine istek yapılarak body'deki data'da istediğimiz kodu çalıştırabiliriz, saldırgan bunu kullanarak sistemi ele geçirebilir
    d. Nostromo web server'ını güncel tutarak bu önlenebilir
- *CVE-2017-11610*
    a. Supervisor'da authenticated user'ın XML-RPC request'leri ile command execution yapabilmesi
    b. Supervisor'un eski sürümleri kullanıldığında ortaya çıkar
    c. Rce ile sistem ele geçirilebilir
    d. Supervisor'un yeni sürümlerini kullanmak