Security for Developers

Ben Kimim?

Faruk Ulutaş

Bilkent Üniversitesi 4. Sınıf Bilgisayar Mühendisliği

https://www.linkedin.com/in/farukulutas/

SecOps Engineer Intern @ Jotform

Full Stack Developer @ BLC-CSS

Cyber Security Enthusiast and Member of PwnLab

İçindekiler

Neden Güvenli Geliştirme Önem Arz Ediyor?

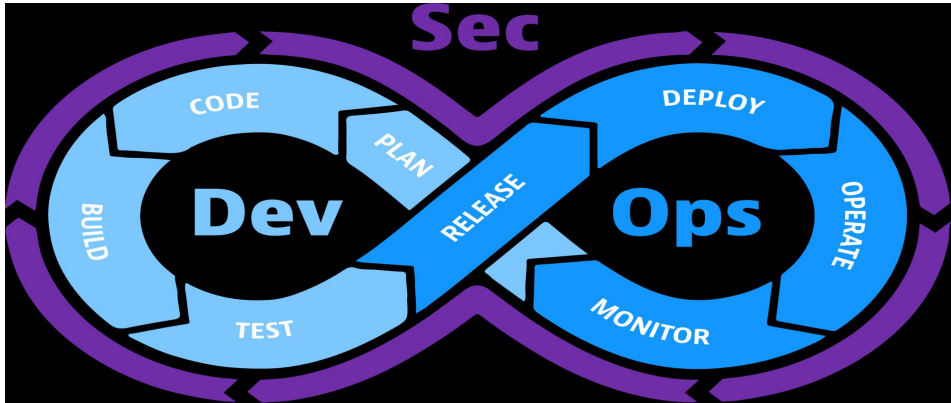
- En hızlı büyüyen siber güvenlik alanı
- 2025’e kadar %164 büyüme (ilan bazında)
- 2019 yılında 4 milyar $ market
- 2025 yılında 15 milyar $ market beklentisi
- Uygulamalar, ürünler internete açık (public)
- Veri Güvenliği ve Gizliliğinin Önemi
- - Yasalar ve Cezalar (Gelir ve İşletme Lisansı Kaybı)
- - Güven ve İtibar Kaybı
- - Verinin Önemi!

Uygulama Geliştirme Güvenliğine Küçük Bir Bakış

- Amaç: Uygulama üzerindeki açıkları bulup düzeltmektir.
- Uygulama sahaya çıkmadan önce yapılmalıdır.
- Hatta, geliştirme sürecine dahil edilmelidir.
- Application Security Test (AST) Çeşitleri:
- - Static Application Security Testing (SAST)
- - Dynamic Application Security Testing (DAST)
- - Interactive Application Security Testing (IAST)
- - Runtime Application Self-Protection (RASP)

Static Application Testing (SAST)

- Genel olarak white-box test
- Uygulama geliştirme sürecinde kaynak kod analizi yapılmasıdır
- CI/CD dediğimiz geliştirme akışını kesintiye uğratmadan bir sonraki aşamadan önce açıkların tespiti ve/veya kapatılmasını sağlar.
- En yaygın kullanımları, SQL injection, buffer overflow, broken authentication gibi sık karşılaşılan OWASP Top 10 zafiyetlerinin tespiti içindir.
- Örn Araçlar: SonarQube, Appknox,
- Coverity, CodeScan, Mend,
- Klocwork, Snyk, Embold, Bandit

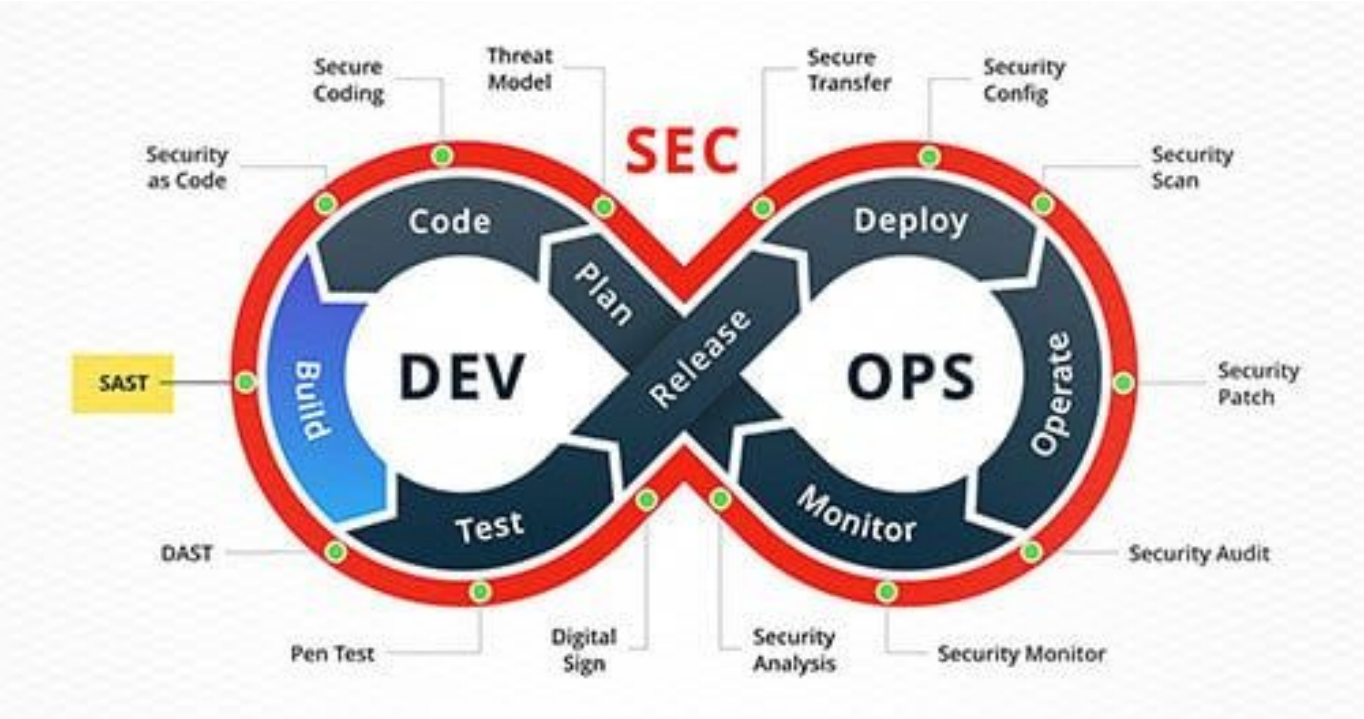


Dynamic Application Security Testing (DAST)

- Uygulama çalışır haldeyken otomatize araçlar ile “easy to explore” dediğimiz bulması kolay olan açıkları aramaktır.
- Böylece çalışma zamanında gelebilecek olası botlardan korunmuş oluruz.
- Örn. açıklar: unauthenticated access, code injection, cross-site scripting (xss).

Interactive Application Security Testing (IAST)

- SAST ve DAST olarak bahsettiğimiz süreçlerin hibrit bir şekilde kullanımıdır.



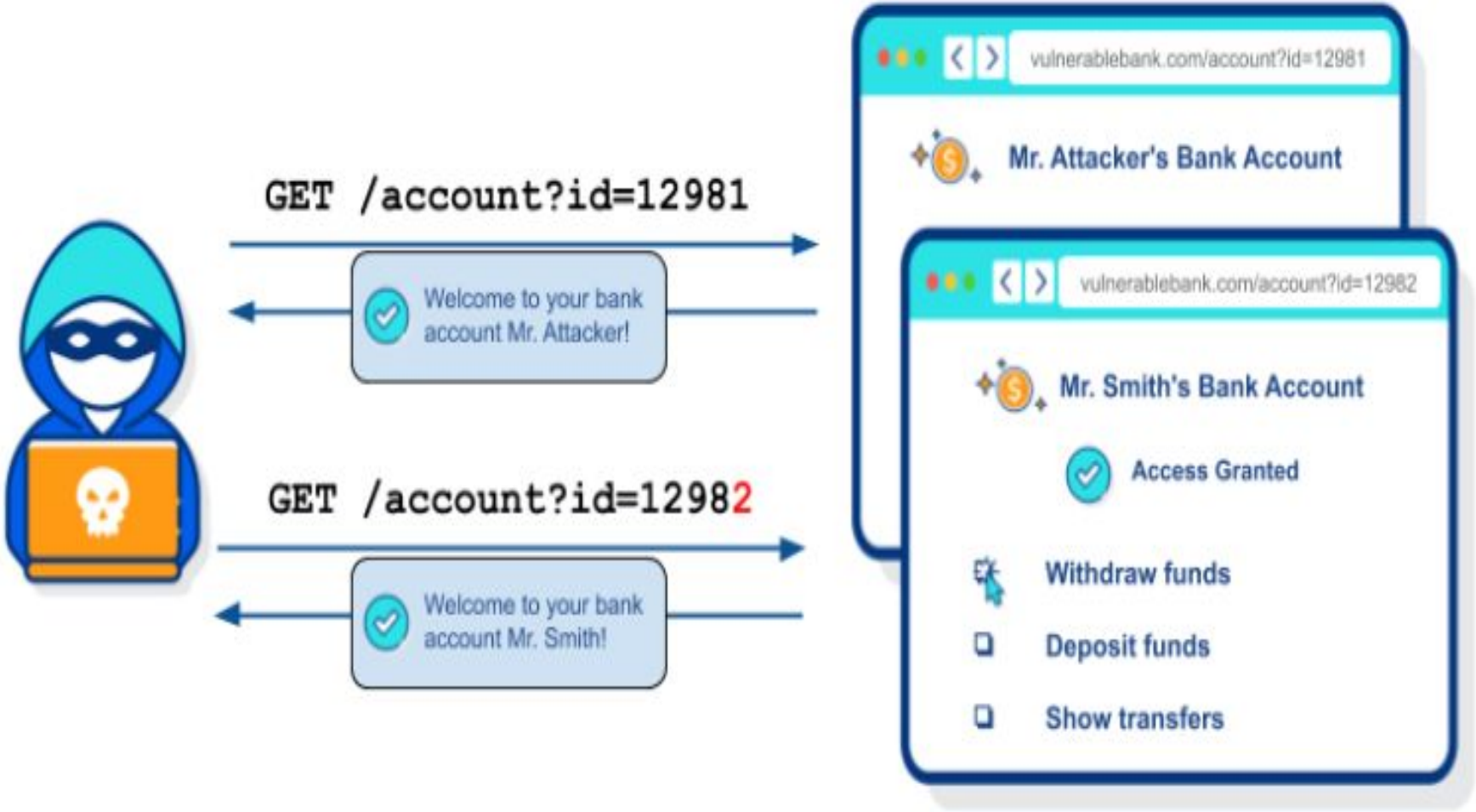
Runtime Application Self-Protection (RASP)

- WAF (Güvenlik Duvarı)'nı ön yüzdeki bir duvar olarak düşünürsek, RASP uygulamayı içten ve dıştan koruyan bir kalkan olarak düşünebiliriz.
- OWASP Top 10 saldırılarına karşı uygulamanızı koruyabilir.
- Güvenlik duvarı ve internet mimarisinde olduğu gibi güncellemelere ihtiyaç duymadan bulut tabanlı her yerde çalışabilir.
- Örn. Araçlar: Fortify, Sqreen, OpenRASP,
- Signal Sciences, Jscrambler, Hdiv, Imperva

A01:2021 – Broken Access Control

- Erişim denetimi, kullanıcıların amaçlanan izinlerinin dışında hareket edemeyecekleri bir politika uygular.
- Hatalar tipik olarak yetkisiz bilgilerin ifşa edilmesine, değiştirilmesine veya tüm verilerin yok edilmesine veya kullanıcının sınırları dışında bir iş işlevinin gerçekleştirilmesine yol açar.
- Vulnerable Example →
- ! /admin/init herkesin erişimine açık olmamalıdır

```
@app.route('/admin/init', methods=['POST'])
def reinitialize():
    cursor.execute("DROP DATABASE analytics")
    return 'Database has been dropped
```


GET /account?id=12981
Welcome to your bank account Mr. Attacker!
GET /account?id=12982
Welcome to your bank account Mr. Smith!
vulnerablebank.com/account?id=12981
Mr. Attacker's Bank Account
vulnerablebank.com/account?id=12982
Mr. Smith's Bank Account
Access Granted
Withdraw funds
Deposit funds
Show transfers

Korunma Yöntemi

- Django “request.user” ile oturum sahibini döner.
- “is_authenticated” ile kullanıcının kimliğini doğrulayıp doğrulamadığını anlayabiliriz. Eğer doğrulamadıysa “AnonymousUser” objesi döner.

```
def admin_init(request):
    if request.user.is_authenticated:
        # Do something for authenticated users.
    else:
        # Do something for anonymous users.
```

Korunma Yöntemi

- Kullanıcılar için izinler oluşturulabilir. Böylece, “User.has_perm()” kullanılarak ilgili metoda erişmek isteyen kişinin izne sahip olup olmadığı kontrol edilebilir.

```
from django.contrib.auth.models import User
from django.shortcuts import get_object_or_404

def admin_init(request, user_id):
    user = get_object_or_404(User, pk=user_id)
    user.has_perm('myapp.change_database')
    ...
```

Korunma Yöntemi

- Flask-Session, bir web uygulamasına sunucu tarafı oturumları için destek ekleyen Flask için bir uzantıdır.
- Flask-Session kullanılarak yine benzer şekilde oturumlar yönetilebilir.

```
from flask import Flask, session, abort
from flask.ext.session import Session

app = Flask(__name__)
app.config.from_object(__name__)
Session(app)

@app.route('/admin/init')
def admin_init():
    if not session.get('user', 'is_authenticated')
        abort(401)
    ...
```

Weak Hashing Algorithm

Hash fonksiyonları matematiksel algoritmalardır.

Tek yönlü ve sabit boyutlu çalışırlar.

```
md5("foo") -> acbd18db4cc2f85cedef654fccc4a4d8
```

```
md5("bar") -> 37b51d194a7513e45b56f6524f2d51f2
```

Zayıf Hash Algoritmalarının kullanımından dolayı sayısız veri ihlali vardır.

Örn: Yahoo'nun 2016'da çaldırdığı bütün kullanıcı bilgileri veri tabanında md5 olarak tutulduğundan sözlük saldırıları ile kırılmıştır.

Zayıf Olduğu Bilinen Algoritmalar

MD5: 90'ların ortalarından beri çarpışma saldırılarına açık olduğu biliniyor ve tamamen bozuk olduğu düşünülüyor.

SHA-1: 2005'ten beri iyi kaynaklara sahip düşmanlara karşı güvensiz olarak kabul edildi ve 2011'de NIST tarafından resmi olarak kullanımdan kaldırıldı.

RIPEMD & RIPEMD-128: 2004'te bildirilen bir çarpışma ile güvensiz olarak kabul edildi.

Whirlpool: 2009'da bir geri tepme saldırısı çarpışmalar sundu.

Mevcut Güvenilir Hash Fonksiyonları

RIPEMD-160/256/320: hepsi sağlam kabul edilse de, farklı güvenlik seviyelerine sahip çoklu varyantlar.

BLAKE2/3: SHA-1/2/3'ten daha hızlı ve uzunluk uzamasına karşı bağışık.

SHA-2: tüm varyantlar çarpışma saldırılarına karşı genel olarak dirençlidir ve çoğu varyant uzunluk uzatma saldırılarına karşı dirençlidir.

SHA-3: SHA serisinin en son yinelemesi; çarpışma ve uzunluk uzatma saldırılarına karşı genel olarak dirençli.

Şifrelerimizi hashlerken ne kullanmalıyız?

bcrypt: Birçok sistemde kullanılan varsayılan parola karma algoritması.

scrypt: bruteforceing'i azaltmak için hashing'i hesaplama açısından yoğun hale getirmek için özel olarak tasarlanmış bir algoritma.

argon2: 2015 Şifre Hash Yarışması'nın galibi; sürecin hesaplama yoğunluğu ince ayar yapılabilir.

PBKDF2: NIST tarafından önerilen bir anahtar türetme algoritması.

A03:2021 - Injection

Uygulamalar kullanıcılardan aldıkları verileri doğrulamadan, temizlemeden veya filtrelemeden kullandıklarında saldırgan kişilerin arka taraftaki işleyişi manipüle ederek istedikleri şeyi sorguyu veya komutu çalıştırması durumudur.

En bilindikleri şu şekilde sıralanabilir: SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection

Bu zafiyetten tespit etmenin en iyi yolu “Kaynak Kod Analizi” yapmaktır.

Kod analizi tarafında SAST, uygulama çalışırken bilinen payloadlar ile DAST analizinin yapılması şiddetle önerilir.

OS Command Injection

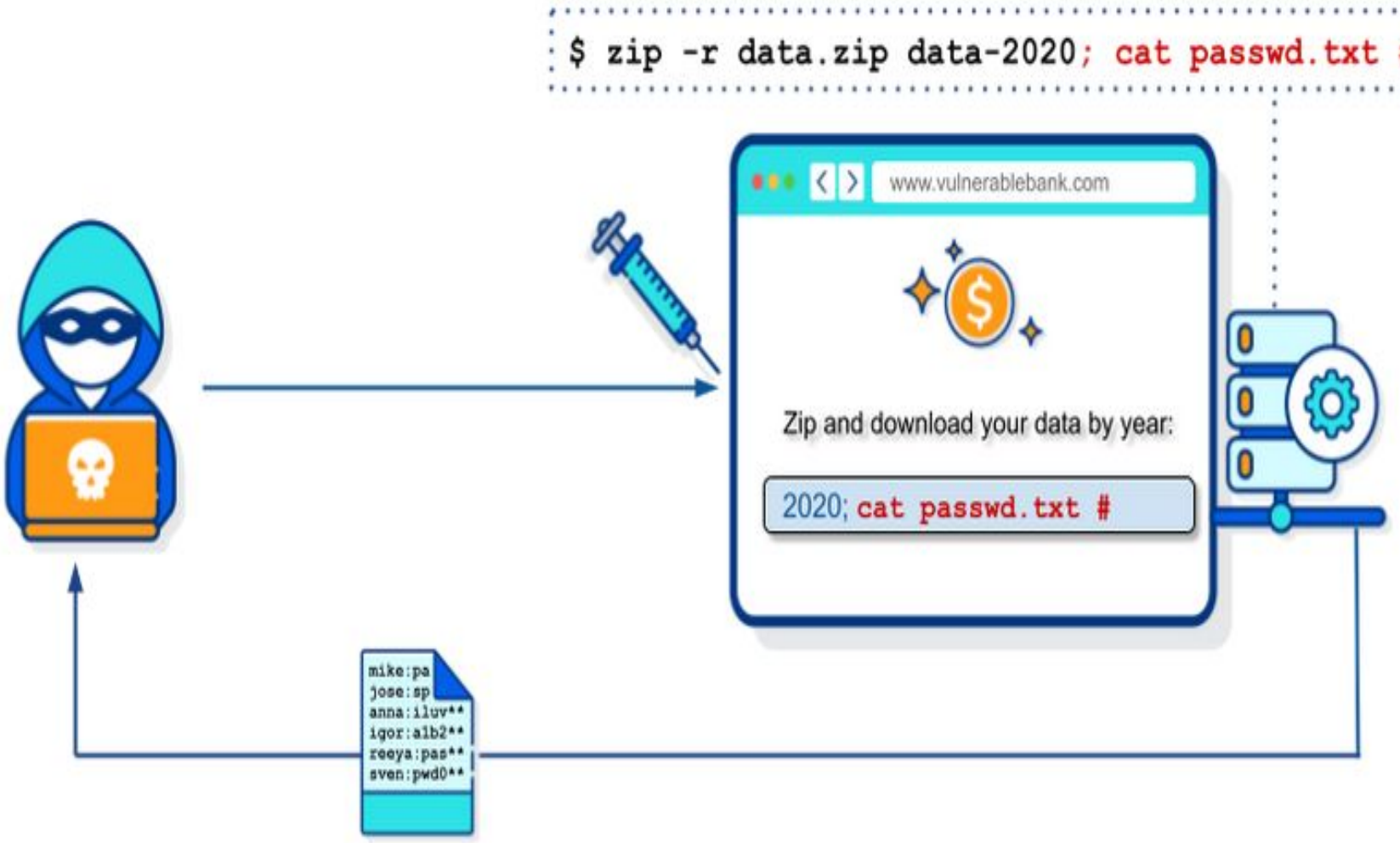
Flask Web uygulaması için yazılmış olan nslookup sorgusunu gerçekleştiren bir kod inceleyeceğiz.

Burada “hostname” direkt olarak gönderilip “cmd” değişkenine eklenip, daha sonra alt bir kabuk parçasında (shell=True) olarak çalıştırıldığı için saldırgan çalıştırmak istedikleri kodları ard arda “;” ile ekleyerek çalıştırabilir.

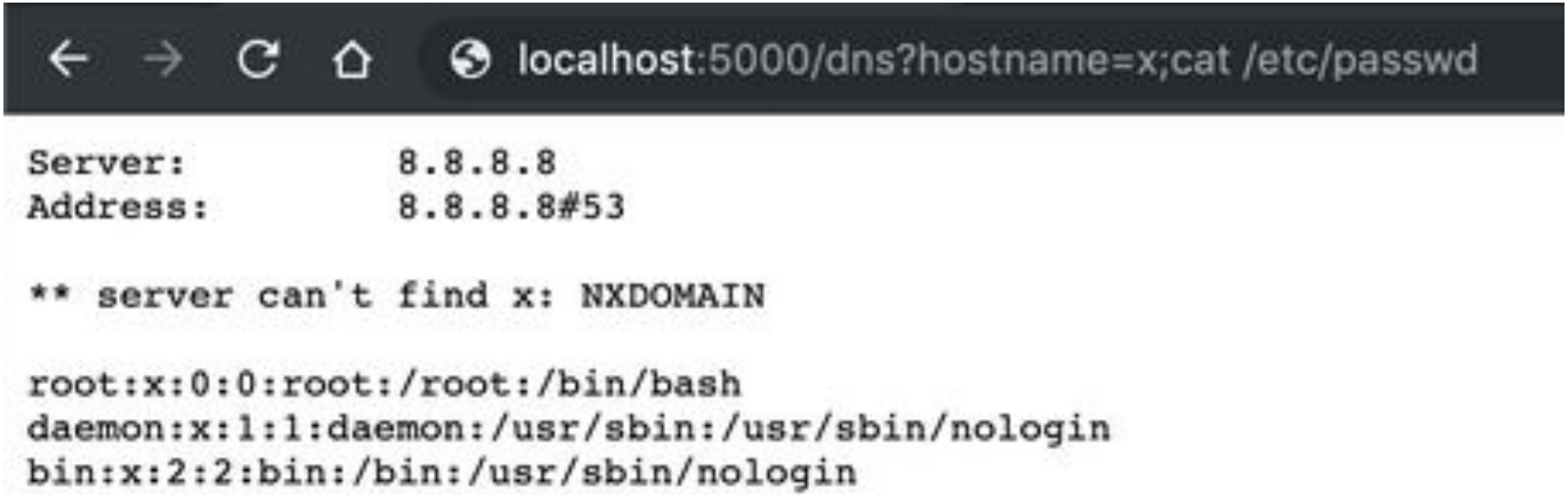
```
@app.route("/dns")
def page():

    hostname = request.values.get(hostname)
    cmd = 'nslookup ' + hostname

    return subprocess.check_output(cmd, shell=True)
```


$ zip -r data.zip data-2020; cat passwd.txt #
www.vulnerablebank.com
Zip and download your data by year:
2020; cat passwd.txt #
mike:pa
jose:sp
anna:iluv**
igor:a1b2**
reeya:pas**
sven:pwd0**

OS Command Injection



Korunma Yöntemi

Python komutları yürütmek için yerel kütüphaneye sahiptir.

OS kütüphanesinde veriler tek bir string olarak filtrelenmeden asla alınmamalıdır.

Örn:

```
os.system(cmd)
os.popen(cmd, ...)
```

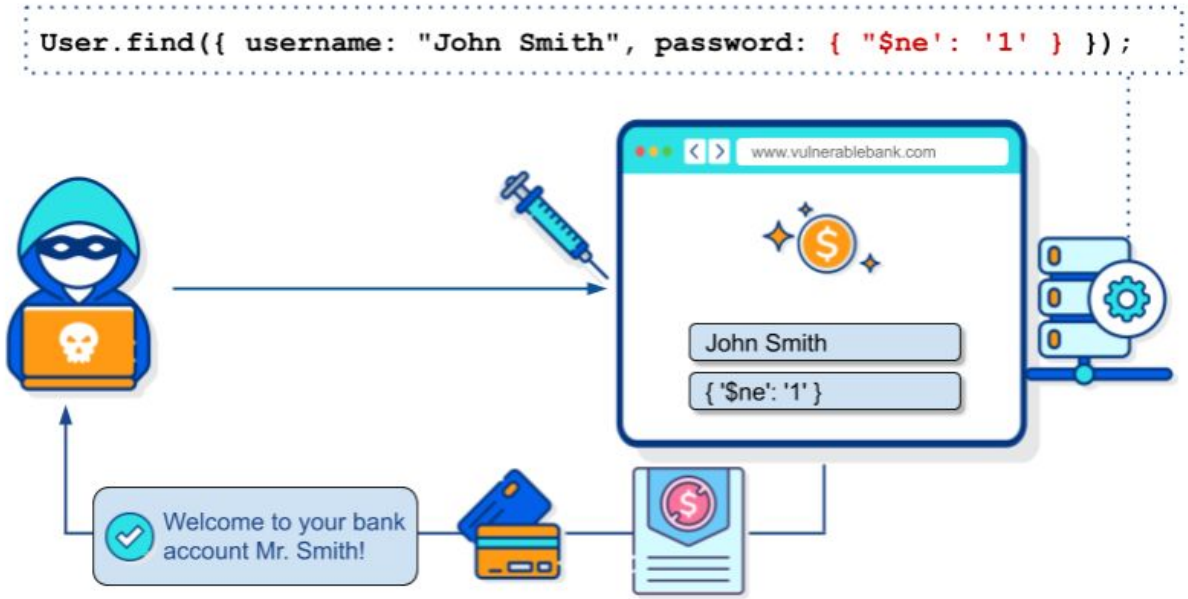
```
subprocess.Popen('nslookup ' + hostname, ... , shell=True) # WRONG
```

Subprocess kütüphanesi kullanılarak, alt kabuk seçeneği False iken, tek bir string yerine parametreli gönderim yapıldığında bu zafiyete karşı korunma yapılmış olunur.

```
subprocess.Popen([ 'nslookup', hostname ], ... , shell=False)
```

NoSQL Injection

Yine benzer şekilde kullanıcının istediği gibi veri girmesine veya arka tarafta gerekli filtreleme ve implementasyonların yapılmış olmasına dikkat etmek gerekir.



NoSQL Injection

Bir web uygulamasının, kullanıcıların kimlik bilgilerini MongoDB'de sakladığını ve giriş sırasında, enjekte edilebilir bir Mongo NoSQL sorgusu kullanarak verilen kullanıcı adı ve parolaya sahip bir kullanıcı olup olmadığını kontrol ettiğini varsayalım.

```
Users.findOne(
  {
    "name" : req.body.name,
    "password" : req.body.password
  }
);
```

```
{
  "name": "user",
  "password": "password"
}
```

NoSQL Injection

$ne (eşit değil) sorgu operatörü, "1" dizesine eşit olmayan bir ada ve parolaya sahip ilk Kullanıcıyı bulmak için Users.findOne() tarafından kullanılacaktır. İfade doğru olduğundan ve her zaman geçerli bir kullanıcı döndürdüğünden, bu istek saldırganın kimlik bilgilerini sağlamadan oturum açmasını sağlar.

```
{
   "name": { "$ne": "1"},
   "password": { "$ne": "1"}
}
```

Korunma Yöntemi

Kullanıcının gönderdiği JSON objesi üzerinde NoSQL anahtar kelimelerinin temizlenmesi ve ilgili dönüşümlerin yapılması gerekir.

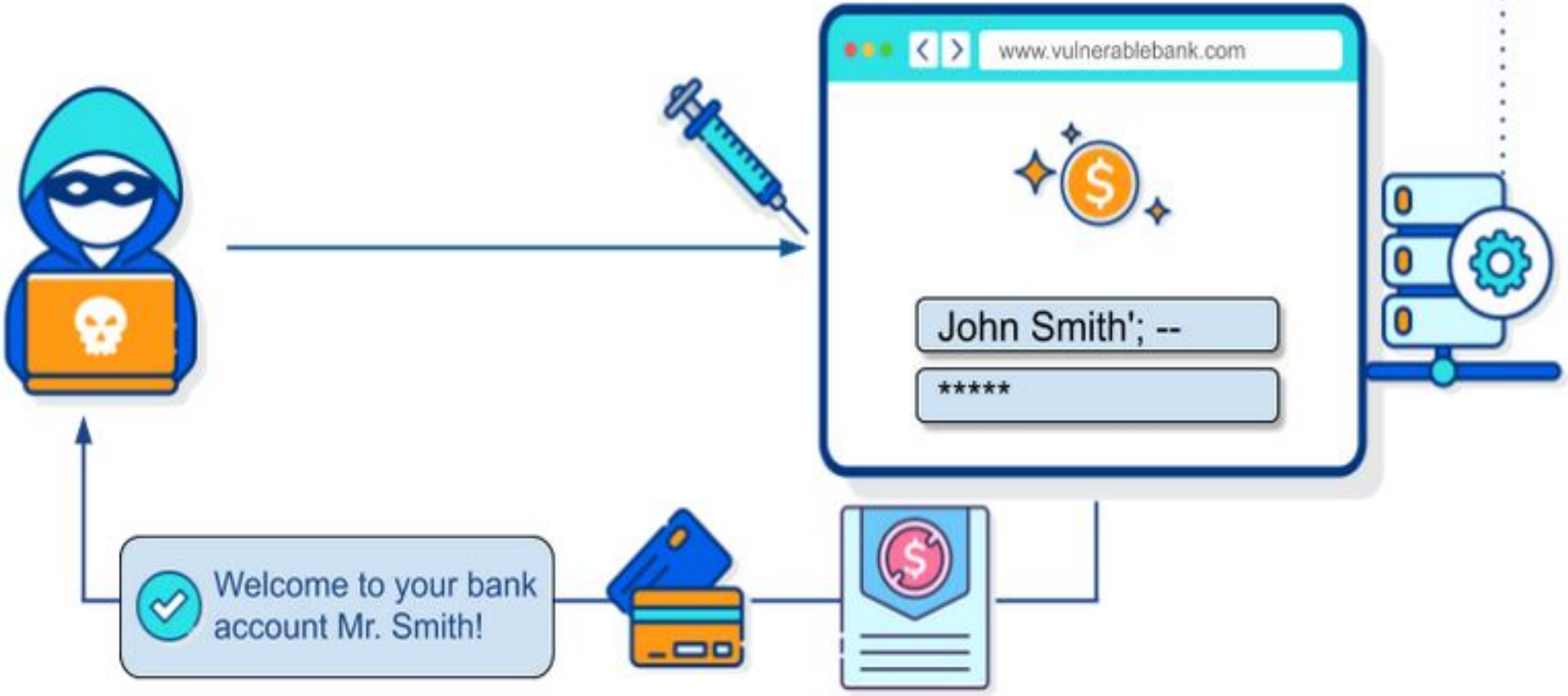
Kullanıcıya hiçbir zaman güvenmemiz gerekir. Bu yüzden, kullanıcıdan alınan verilerin çalıştığı veri tabanında yetkilendirmelerin minimum olması gerekir.

SQL Injection

Uygulamalar, kullanıcı tarafından sağlanan veriler doğrulanmadığında, kaçış karakterleri için filtrelenmediğinde veya uygulama tarafından temizlenmediğinde saldırılara karşı savunmasızdır.

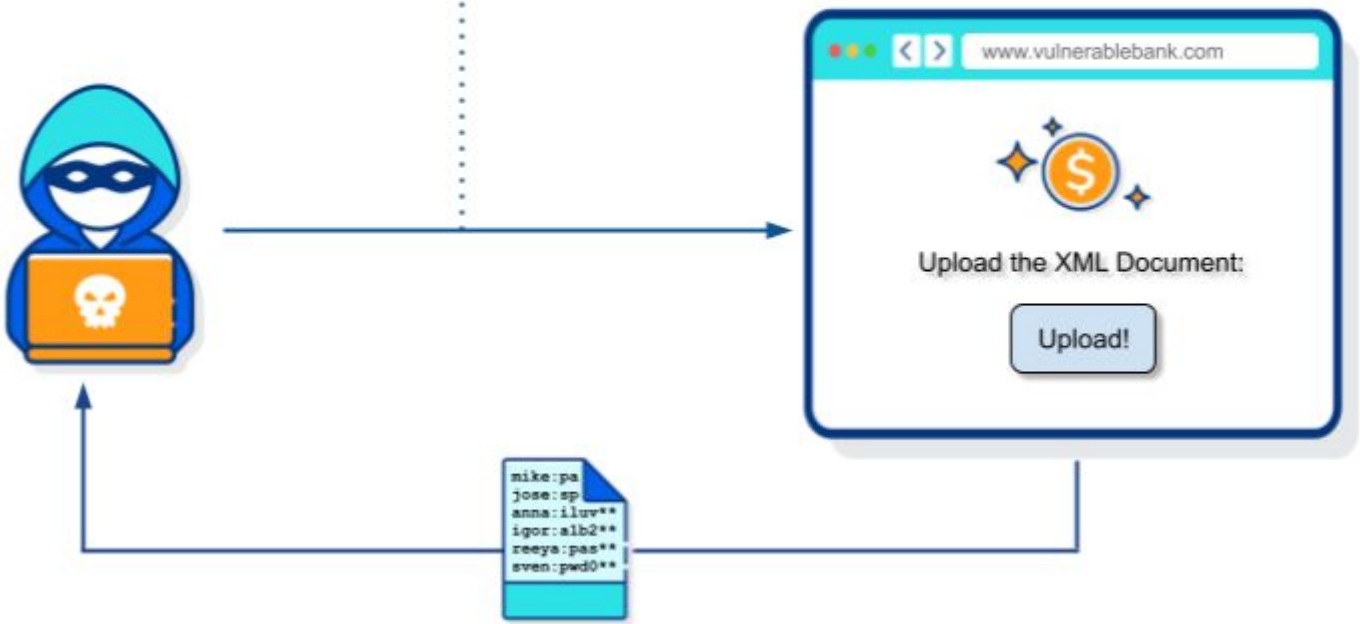
Saldırgan, istemciden uygulamaya giriş verileri aracılığıyla bir SQL sorgusunu işlemek için SQL Injection'ı kullanabilir, böylece SQL sunucusunu güvenilmeyen girdi kullanılarak oluşturulan istenmeyen bir işlemi yürütmeye zorlayabilir.

```
SELECT * FROM users WHERE name='John Smith'; --' and password='wrong'
```



XML Injection

```
<!DOCTYPE catalog[ <!ENTITY xxe SYSTEM "file:///data/passwd.txt"> ]>
```



Unrestricted File Download

assetName kullanıcı tarafından kontrol ediliyorsa, örneğin dosya olarak ../../../../../../etc/passwd kullanarak bir “directory traversal” gerçekleştirmek ve amaçlanan dizinden kaçmak mümkündür.

Bu, işletim sistemi tarafından /etc/passwd olarak

standartlaştırılan

/opt/wwwdata/assets/../../../../../etc/passwd

yolu ile sonuçlanır.

```
using System;

public class Assets
{
    public static string assetsDir = "/opt/wwwdata/assets/";

    public static string getAssetPath(string assetName)
    {
        return System.IO.Path.Combine(
            assetsDir,
            assetName
        );
    }
}
```

Korunma Yöntemi

Bir yolun istenmeyen bir konuma işaret etmediğini doğrulamak için, mutlak yolu almak için Path.GetFullPath() öğesini kullanın ve ardından beklendiği gibi başladığını doğrulayın.

```
public static string getAssetPath(string assetName)
{
   string assetPath = System.IO.Path.Combine(
      assetsDir,
      assetName
   );

   if(!Path.GetFullPath(assetPath).StartsWith(assetsDir)) {
      throw new System.InvalidOperationException("Provided asset name is invalid");
   }

   return assetPath;
}
```

Korunma Yöntemi

Yolu temizlemek için, kullanıcı tarafından sağlanan dosya adının yalnızca son kısmını birleştirmek için System.IO.Path.GetFileName() kullanmak mümkündür. Bunun sonucunda, az önceki atak “/opt/wwwdata/assets/passwd” ile sonuçlanır.

```
public static string getAssetPath(string assetName)
{
    return System.IO.Path.Combine(
        assetsDir,
        System.IO.Path.GetFileName(assetName)
    );
}
```

Kaynakça

https://knowledge-base.secureflag.com/vulnerabilities/broken_authentication/broken_authentication_category.html

https://securityintelligence.com/articles/why-demand-for-application-development-security-skills-exploding/

https://owasp.org/Top10/

https://owasp.org/www-community/Source_Code_Analysis_Tools

https://www.perforce.com/blog/kw/application-security-development-best-practices

https://crashtest-security.com/importance-appsec-basics/#application-security-reports

https://www.vmware.com/topics/glossary/content/application-security.html

https://www.mend.io/resources/blog/dast-dynamic-application-security-testing/

https://geekflare.com/rasp-tools/

https://www.appsealing.com/rasp-security-runtime-application-self-protection/

Beni dinlediğiniz için teşekkürler.

İletişim: https://www.linkedin.com/in/farukulutas/

Siber güvenlik alanında daha fazla bilgi edinmek için: https://pwnlab.me/

Mini-CTF deneyimi için: https://ctf.pwnlab.me/

Bizi takip etmeyi unutmayın!

Twitter: https://twitter.com/PwnLabMe (PwnLab.Me)

Instagram: https://www.instagram.com/pwnlab.me/ (pwnlab.me)

Telegram: https://t.me/CyberSec_TR (Cyber Security | Turkey)