

# Dangers of Web Applications

Jack Mannino



# Topics

- ▶ The Big Picture
- ▶ The Risks
- ▶ A New Way of Looking At Threats
- ▶ The Right Approach
- ▶ Resources

# The Big Picture

## Priority One: Client-side software that remains unpatched.

Waves of targeted email attacks, often called **spear phishing** are exploiting client-side vulnerabilities in commonly used programs such as Adobe PDF Reader, QuickTime, Adobe Flash and Microsoft Office. This is currently the primary initial infection vector used to compromise computers that have Internet access. Those same client-side vulnerabilities are exploited by attackers when users visit infected web sites. (See Priority Two below for how they compromise the web sites). Because the visitors **feel safe** downloading documents from the **trusted sites** they are easily fooled into opening documents and music and video that exploit client-side vulnerabilities. Some exploits do not even require the user to open documents. Simply accessing an **infected website** is all that is needed to compromise the client software. The victims' infected computers are then used to propagate the infection and compromise other internal computers and sensitive servers incorrectly thought to be protected from unauthorized access by external entities. In many cases, the ultimate goal of the attacker is to **steal data** from the target organizations and also to **install back doors** through which the attackers can return for further exploitation. On average, major organizations take at least **twice as long to patch client-side vulnerabilities as they take to patch operating system vulnerabilities.** In other words the highest priority risk is getting less attention than the lower priority risk.

## Priority Two: Internet-facing web sites that are vulnerable.

Attacks against web applications constitute **more than 60%** of the total attack attempts observed on the Internet. These vulnerabilities are being exploited widely to convert trusted web sites into malicious websites serving content that contains client-side exploits. Web application vulnerabilities such as SQL injection and Cross-Site Scripting flaws in open-source as well as custom-built applications account for **more than 80%** of the vulnerabilities being discovered. Despite the enormous number of attacks and despite widespread publicity about these vulnerabilities, most web site owners fail to scan effectively for the common flaws and become unwitting tools used by criminals to infect the visitors that trusted those sites to provide a safe web experience.

## SANS Top Cyber Security Risks

<http://www.sans.org/top-cyber-security-risks/#trends>

# The Big Picture

## How Did We Get Here?

- ▶ Organizations started patching and using firewalls
- ▶ Number of operating system vulnerabilities dropped
- ▶ Patching, blocking services at the firewall, and depending on Intrusion Prevention became standard
- ▶ As these became the “security standard”, emerging areas of risk were ignored
- ▶ Attackers are smart, and began taking the path of least resistance.....

# The Big Picture

## Web Applications are EVERYWHERE

- ▶ The days of traditional Web Server → Database architectures are long gone
- ▶ We trust the websites we use every day, right?
- ▶ We have web applications:
  - In our pockets  
Mobile apps that communicate with web services
  - On our desktops  
Adobe AIR allows file system IO  
Client-side storage (SQL databases)
  - In the “cloud”  
Think anything and everything Google does



# The Big Picture

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security <u>Misconfiguration</u> (NEW)
A10 – Failure to Restrict URL Access	A7 – Failure to Restrict URL Access
<not in T10 2007>	A8 – <u>Unvalidated</u> Redirects and Forwards (NEW)
A8 – Insecure Cryptographic Storage	A9 – Insecure Cryptographic Storage
A9 – Insecure Communications	A10 - Insufficient Transport Layer Protection
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

# The Risks

## What Are The “Risks”?

- ▶ Data Loss
  - Personally Identifiable Information (PII)
  - Critical Operational Information
- ▶ Complete System Compromise
  - Underlying web servers and databases
  - Attached systems
- ▶ Using Web Attacks as Pivot Points
  - Many attacks do not require missing patches
  - Weaknesses in browsers allow intranet attacks
- ▶ Firewalls and Perimeter Security Are Ineffective
  - Unless you are inspecting SSL traffic, you are blind
  - Complex attacks can originate on external domains

# The Risks

## Cross Site Request Forgery (CSRF)

- ▶ Commonly referred to as CSRF (“Sea Surf”)
- ▶ Browser is designed to send cookies for a given domain whenever they exist
- ▶ If a resource for Domain A is requested from Domain B, the cookies (authentication information) will be sent for any requests for Domain A
- ▶ This allows an attacker to perform authenticated transactions transparently
- ▶ Why is this dangerous?



# The Risks

## Cross Site Request Forgery (CSRF)



Referenced from an OWASP presentation

[http://www.owasp.org/images/a/a1/AppSec\\_DC\\_2009\\_-\\_OWASP\\_Top\\_10\\_-\\_2010\\_rc1.pptx](http://www.owasp.org/images/a/a1/AppSec_DC_2009_-_OWASP_Top_10_-_2010_rc1.pptx)

# The Risks

## Broken Authentication and Session Management

- ▶ Essentially opens the floodgates....
- ▶ More than just weak passwords and absence of lockout policies
- ▶ Session tokens are often predictable and have excessively long lifetimes
- ▶ When user privileges are not enforced through a session token, escalation to higher access rights (administrator) or access to other user accounts becomes possible

# The Risks

## Broken Authentication and Session Management

### Example

- User A is assigned a session token with the value “AuthToken=XYS3refg4i287jf9235”
- The token has a lifetime of 12 hours
- An attacker is able to register his/her own account
- The attacker attempts several thousand logins, and notices that the session token increments in a predictable way (first 15 characters are random, but the last 4 increment by 1 each time)
- The attacker can now access authenticated accounts by guessing valid session tokens, some of which may grant the attacker full administrative rights

# The Risks

## Injection Attacks

- ▶ Most common and prevalent is SQL Injection
- ▶ SQL Injection allows an external attacker to interface directly with your backend database
- ▶ May result in loss of information, unauthorized data manipulation, or a complete compromise of the underlying servers
- ▶ Other injection attacks include LDAP Injection, SMTP Injection, and XML Injection

# The Risks SQL Injection

← → ↻ 🏠 ☆ http://hackme.ntobjectives.com/sql\_inject/login\_get.php



NT OBJECTIVES,  
INCORPORATED



HACKME TEST SITE

[Home](#) [SQL Injection](#) [XSS](#) [HTTP Res Splitting](#) [Reset Session](#) [View Source](#)

## Please Login


Username:

Password:


Copyright © NT OBJECTives, Inc. All Rights Reserved.

# The Risks SQL Injection

http://hackme.ntobjectives.com/sql\_inject/login\_get.php?username=admin'+and+1%3D1--+&password=&Submit=Login



NT OBJECTIVES,  
INCORPORATED



HACKME TEST SITE

[Home](#) [SQL Injection](#) [XSS](#) [HTTP Res Splitting](#) [Reset Session](#) [View Source](#)

**Hackme Helper Window**  
Query: SELECT \* FROM accounts WHERE username='admin' and 1=1-- ' and password =  
'd41d8cd98f00b204e9800998ecf8427e'

**Welcome admin**

whatever page content...

Copyright © NT OBJECTives, Inc. All Rights Reserved.



# The Risks

## SQL Injection

### How did this happen?

- ▶ Web application did not properly filter input
- ▶ Web application did not use “parameterized queries” for database calls
- ▶ The string ‘ or 1=1-- forces the database to always return a true value
- ▶ As a result, it was possible to bypass the login page

# The Risks

## SQL Injection

**What else could we have potentially done?**

- ▶ Dropped a database table
- ▶ Executed system commands (imagine having your own command line on that server)
- ▶ Injected malicious code to serve to users
- ▶ Tampered with information contained in the database

# The Risks

## Cross Site Scripting (XSS)

- ▶ Allows an attacker to run arbitrary Javascript and HTML in a victim's browser
- ▶ Can be stored, reflected, or Document Object Model (DOM)–based
- ▶ Used for more than creating popup windows!

# The Risks

## Cross Site Scripting (XSS)

- ▶ Javascript can be used to execute shellcode within a browser
- ▶ What you see isn't always what's really there!
- ▶ Information can be sent across domains
  - Session tokens
  - User information
- ▶ Launch attacks from a victim's browser using HTTP and other protocols (IRC, SSH, FTP, etc)

# A New Way Of Looking At Threats

## Vulnerability Chaining

- ▶ Moderate Threat + Moderate Threat + Low Threat = Really Bad
- ▶ Some attacks alone may require a significant level of time and skill, but with the presence of other issues they become trivial to exploit

# A New Way Of Looking At Threats

## Vulnerability Chaining

### Example

- ▶ An XSS vulnerability exists in the “Update Profile” portion of an application
- ▶ An attacker wants to get this XSS to execute for another user
- ▶ Several options exist
  - Find SQL Injection
  - Brute-force attack their password
- ▶ Both of those options may take a significant amount of time and may set off alarms
- ▶ Solution– Attack broken authentication!



# A New Way Of Looking At Threats

## Vulnerability Chaining

- ▶ Many organizations classify vulnerabilities differently according to the authentication level required to perform it
- ▶ As an example, the DoD rates XSS and SQL Injection as moderate vulnerabilities if they are discovered in the authenticated portion of an application, but critical if it requires prior authentication
- ▶ If authentication is trivial to circumvent, shouldn't this mean we treat the vulnerability differently?

# A New Way Of Looking At Threats

## Pretend The Firewall Doesn't Exist

- ▶ There are many, many ways to get inside the perimeter
- ▶ XSS Shells, Social Networking “Bots”, and many more
- ▶ Flash is dangerous...very dangerous.
- ▶ Assume that your firewall has already been breached...now what?

# The Right Approach

- An application that contains no “sensitive” data shouldn’t be ignored
- Identify the threat vectors that are relevant to an application or system
- Develop a security plan and model it according to these threats

# The Right Approach

- Scanning can help quickly find “low hanging fruit”, but shouldn’t be where you stop
- Start thinking about security early in an application’s development– in production is not the right time
- Code review and testing in a runtime environment each have unique advantages
- One should not replace the other, rather they should complement each other

# The Right Approach

## Source Code Analysis (SCA)

- ▶ Direct access to code
- ▶ Easier to determine root cause of issues
- ▶ Unobstructed view of actual vulnerabilities
- ▶ Automated SCA tools can return millions of false positives
- ▶ Many issues can be found faster in a runtime environment
- ▶ Tools cannot detect logic or contextual issues

Strengths

Weaknesses

# The Right Approach

## Runtime Analysis

- ▶ A single tested parameter may represent hundreds of lines of code
- ▶ Wider choice of commercial and open-source tools
- ▶ Represents an actual attacker's view
- ▶ Best way to test how multiple browsers handle an application
- ▶ Manual Runtime Analysis is excellent for discovering logic and contextual issues
- ▶ Some holes are difficult to detect without access to code
- ▶ Automated tools have known issues testing sites using Ajax and Flash
- ▶ Automated tools lack the intelligence to understand and subvert filters

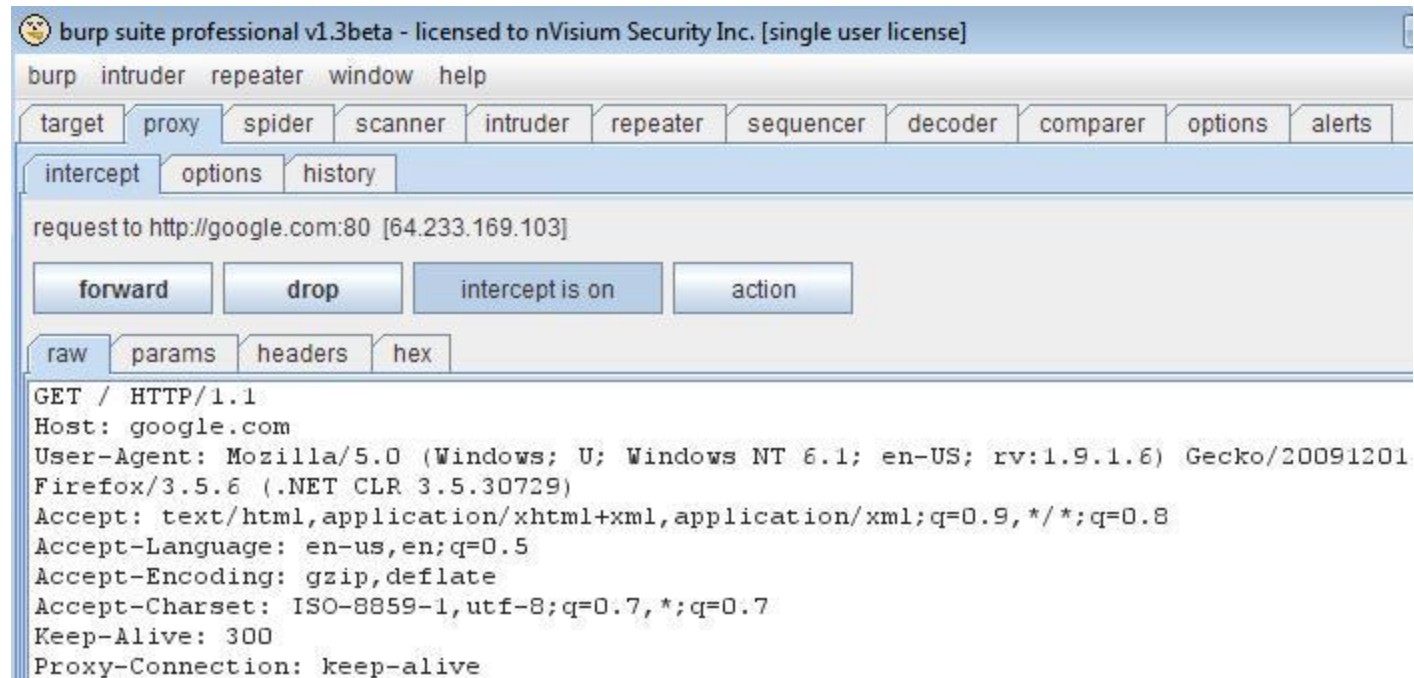
Strengths

Weaknesses



# The Right Approach

## Runtime Analysis Tools



Screenshot of Burp Suite Pro

# The Right Approach

## Summary

- ▶ It is critical to understand the limitations of automated security tools
- ▶ Automation is best used to detect “low hanging fruit”
- ▶ Manual testing is resource intensive and requires intimate knowledge of web technologies
- ▶ Repeatable methodologies are essential to establishing an efficient assessment program

# Resources

- ▶ OWASP Top 10  
[http://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
- ▶ Web Application Security Consortium  
<http://www.webappsec.org/>
- ▶ OWASP Phoenix Project  
<http://www.owasp.org/index.php/Phoenix/Tools>
- ▶ Threat Modeling  
[http://www.owasp.org/index.php/Threat\\_Risk\\_Modeling](http://www.owasp.org/index.php/Threat_Risk_Modeling)
- ▶ SANS Top Cyber-Security Risks  
<http://www.sans.org/top-cyber-security-risks/#trends>

# Questions?

## Contact Information:

Jack Mannino

[jack@nvisiumsecurity.com](mailto:jack@nvisiumsecurity.com)