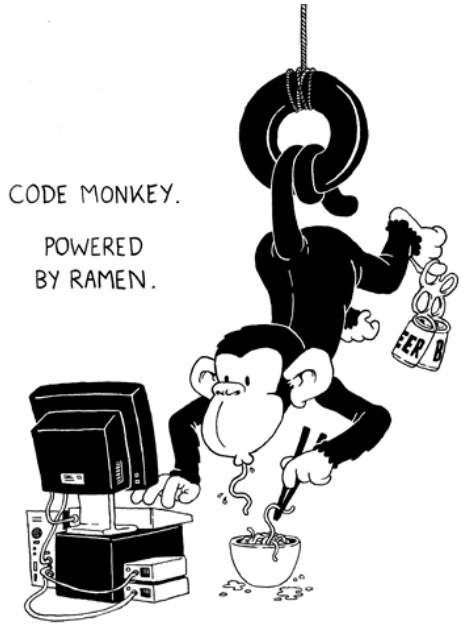
Web Application Attacks

The whats, hows, and OMGs!!!

Who Am I

- Code Monkey
  - Write a lot of software
  - Write in a log of languages
- Security Aficionado
  - Study Hacks and Hacking
  - Study how software is abused
  - Practice hacking techniques



Web Applications

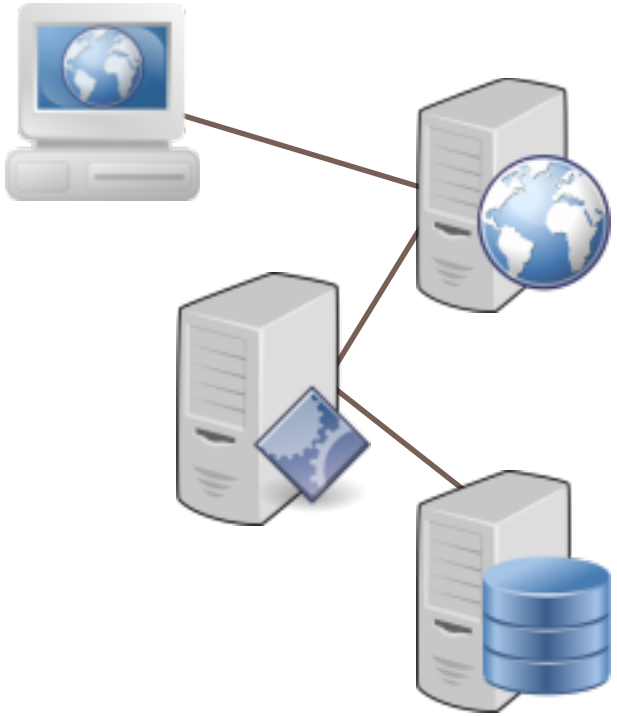
What are they?

How are they different from desktop applications?

How do attackers think about them?

Web Applications

- 3-tiered architecture
  - Web Server
  - Application Server
  - Database Server
- Multiple attack surfaces
  - Don’t forget the client (browser)



Web Server

- Handle client requests
- Serve up web pages via HTTP/HTTPS
- Comes in many flavors:
  - Apache
  - IIS
  - Nginx
  - Lighttpd
  - And many, many, many more… (https://en.wikipedia.org/wiki/Comparison_of_web_server_software)

Application Server

- Does the computational work of creating and serving up a web page.
- Many different languages:
  - Java
  - ASP.NET
  - Python
  - Go
  - PHP
  - And many, many more… Seriously, a lot more (https://en.wikipedia.org/wiki/Comparison_of_application_servers)

Database Server

- Holds the **Gold**
- Stores and processes data
- SQL Servers
  - MS SQL
  - MySQL
  - PostgreSQL
  - Oracle
  - And More...
- NoSQL Server
  - MongoDB
  - Cassandra
  - And More...

Put on Your Hacker Hats

OWASP Top 10

A1-Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2-Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4-Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5-Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6-Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7-Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8-Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10-Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Attack Types

- Server Side
  - Injection
  - Cross-site Request Forgery (CSRF)
- Client Side
  - Cross-site Scripting (XSS)
- Hybrid
  - Combinations of above

Injection Attacks

- SQL Injection
- Command Injection
- Code Injection

SQL Injection

SQL Injection attacks leverage **unvalidated** user input mechanisms to `**inject**` malicious SQL code inside of an executed query. These attacks are often used to bypass improperly configured authentication, or to retrieve privileged information from the underlying database.

WAIT… WTH is SQL?

- Language of databases
- Structured Query Language
- Used by Relation Database Management Systems (RDMS)
- Particular RDMS provide specific functionality.
- General SQL can be used for most queries

SQL Query Verbs

- **SELECT** – Retrieve data from a table
- **INSERT** – Add data to a table
- **UPDATE** – Modify existing data in a table
- **DELETE** – Delete data from a table
- **DROP** – Delete an entire table
- **UNION** – Combine data from multiple queries

SQL Query Modifiers

- **WHERE** – Filter SQL query by specified condition
- **AND / OR** – Combine with WHERE to narrow the SQL Query
- **LIMIT <start>, <count>** – Limit number and which rows are returned
- **ORDER BY** - Sort by specified columns

How Queries Work

SELECT id, name **FROM** products **WHERE** price >= 10.00;

SELECT id, name **FROM** products **WHERE** name **LIKE** '%cup%';

SELECT id, name **FROM** users **WHERE** password = 'sooper$ecret';

SELECT id, name **FROM** users **WHERE** name = 'john' **OR** name = 'george';

Back to the Attack

Think about a login form. It might use a query to check if the user is in the database and has the right password.

```
SELECT * FROM users WHERE username = '$user' AND password = '$pass';
```


```
SELECT * FROM users WHERE username = 'admin' AND password = '' OR 'a'='a';
```

$user = admin
$pass = ' OR 'a'='a

But Wait… There's More

- Blind SQL Injection
  - Error Based
  - Boolean Based
  - Timing Based

Command Injection

Command Injection attacks leverage **unvalidated** user input mechanisms to **`inject`** system commands that are executed on the web server. These attacks are often used to gain a foothold on a system.

Command Injection

- Leverage system calls already in code
  - the magic characters
    - &, &&, ||, <, >, ;

Command Injection

- Example:
  - New accounts require a directory to store configs
  - Application accepts a username parameter
  - It then runs `mkdir username`

- Attacker supplies username ‘john**; rm-rf /**’

Code Injection

Code injection attacks are attacks which consist of injecting code that is then interpreted/executed by the application. These attacks take advantage of **improperly validated** user/data input.

Code Injection

- Take advantage of input executed as code
  - php / javascript - eval()
  - Python - exec
- File Upload
  - With the help of Local File Inclusion

Cross-Site Request Forgery

Cross-site request forgery is an attack that forces a user to execute unwanted actions on a web application that they are authenticated with. Can be combined with cross-site scripting attacks to force unsuspecting users to execute malicious action.

```
http://bank.com/transfer.do?acct=MARIA&amount=100000
```

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is a form of **injection** attack, which enables attackers to inject client-side script into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin policy.

Javascript

Javascript is the language of the client (browser). It is code that is provided with a web page and is executed on the client computer.

Cross-Site Scripting (XSS)

Three Flavors

- Non-persistent
- Persistent
- Dom-based

Non-persistent (XSS)

Also known as reflected cross-site script attacks, these attacks take advantage of sites that include request input in the response. These attacks are usually performed by getting a user to click a link with a malicious script included in URL GET parameter.

Persistent (XSS)

Also known as stored cross-site scripting attacks. These attacks leverage the web applications storage medium (**database**) to store and replay the malicious script to users of the site.

Dom-based (XSS)

This version of cross-site scripting takes input and alters the document object model (DOM) of the site to inject malicious javascript. This differs from other forms of cross-site scripting by not being directly part of the response from the server, rather being entirely handled in the client browser.

Demo Time

Unless there are burning questions...

Mutillidae

- An Intentionally vulnerable set of web applications
- Based on OWASP Top 10
- Runs on LAMP / WAMP / XAMPP
- Project Page:
  - http://sourceforge.net/projects/mutillidae/
    (http://goo.gl/XbLBfs)

Thank You

Joshua Barone

email: joshua.barone@gmail.com

twitter: @tygarsai

blog: http://caveconfessions.com

slides: http://goo.gl/B0Z13j