WEBAPP SECURITY

CHEATING TO WIN

James McMurry
CEO
Milton Security Group
@jmcmurry

Lee Neely
Senior CyberAnalyst
Lawrence Livermore Laboratory
@lelandneely
Introduction 15 minutes

Target Audience: You know of Kali, You know of OWASP, You know what a Web App is

System requirements: None. There is a notebook in front of you to be shared by two people. Find a new friend!

Tools provided: Laptop running Kali Linux with the Lab tools and scripts preinstalled using Docker

Lab 1 Password Fuzzing with ZAP: 25 minutes

Lab 2 SQL Injection: 25 minutes

Lab 3 Passive Recon: 25 minutes

Lab 4 ACTIVE Recon AND EXPLOITATION (CHEATING): 25 minutes
Cheating To Win !!!

- **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
Cheating To Win !!!

- 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

*Cheat : Recon to Exploitation in seconds. Using AutoSploit.*
You know this: Web App Security Overview

It’s harder than it looks. You need to identify:

- Critical assets of the organization
- Genuine users who may access the data
- Level of access provided to each user
- Various vulnerabilities that may exist in the application
- Data criticality and risk analysis on data exposure
- Appropriate remediation measures
You know the problems:

- Because the browser is under the control of the user.
  - Arbitrary data could be submitted
  - Alteration of parameters, cookies, etc. happens
  - Client controls are out of your control
  - All input must be considered malicious
- Tools are everywhere, those tools you use for testing are the tools used to hack you.
The top 5 common vulnerabilities are:
- Injection vulnerabilities
- Cross-site scripting
- Broken authentication and session management
- Insecure direct object references
- Security misconfiguration
Ground Rules

BE NICE, NO ATTACKING EACH OTHER!

ONLY ATTACK TARGETS WE TELL YOU TO!

If you’re not comfortable changing your system, don’t!
Getting Started

In front of you is a notebook with all the tools pre-installed and ready to go.

Please pair up: 2 people per system.

Everything is connected to the class network.

If there is an issue, you can rollback to original configuration by running:

Sudo rollback.sh

Don’t Die
LAB 1 - PASSWORD FUZZING WITH ZAP!
Sometimes Web Sites include information about the company that users incorporate into passwords.

We can spider the site, collecting keywords to build a wordlist.

Then we take the wordlist and try them as passwords for either known or standard accounts, such as admin.

We could do this manually, but tools make it much easier.
Tools for this lab:

- CeWL - from Robin Wood @dijininja - command line tool that harvests keywords
- Zed Attack Proxy (ZAP) from OWASP - proxy that we will use to try password combinations to access the accounts.
- Dojo-basic - vulnerable website for classroom use
Lab 1 - Goals - Password fuzzing using ZAP

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 Fuzzer to fuzz the login password for "john"
- Use the john.txt list from FuzzDB
Use CeWL to crawl web site finding keywords

Configure ZAP to use those words to build passwords for the admin account

Watch the size of the returned data to find success

Repeat for the John account
Lab 1 - Using CeWL to build a wordlist

- **Syntax**: `cewl [options] <target>`
- Review the CeWL options
  - `-d`: depth
  - `-w`: write to (dictionary) file
  - `-e`: e-mail addresses
  - `-a`: metadata...
- Create a wordlist from Dojo-Basic
  - `cewl -a -w /tmp/words.txt http://dojo-basic`
Lab 1- ZAP setup: Add URLs to history

In Chrome, with ZAP selected as proxy, browse to http://dojo-basic/

- Click the Login link
- Enter ZAP for username
- Enter ZAP for password
- Click Submit

**Bad user name or password!**
Fuzz Admin Login

- Tools - Fuzz
- Message Type: HTTP
- Select the login script with username/password
Fuzz Admin Login

1) Set the user to Admin:
   - Highlight “ZAP” after user_name
   - Click Add to add a fuzz location
   - Click Add to add a payload of type strings, value admin
   - Click add, then click OK
Fuzz Admin Password

2) Highlight “ZAP” after password
   - Click Add to add a fuzz location
   - Click Add to add a payload of type file select your cewl wordlist (/tmp/cewl.txt)
   - Click Add, Click OK
Fuzz!
Click Start Fuzzer
Wait for the progress bar
Look at the response header size - successful login is different

Select that task - notice the payload, or request shows the value sent.

Your Turn:

- User: john
- Wordlist: /pentest/fuzzers/fuzzdb/wordlists-user-passwd/passwds/john.txt
LAB 2 - SQL INJECTION
SQL Injection first discussed about 1998.

SQL Injection also known as SQLi.

SQLi is an attack technique against data-driven applications.
Nefarious SQL statements are inserted into fields which then are executed against the database.

SQLi works when the input is not properly filtered or not strongly typed.

SQLi can be used to not only access and/or change information in a database, but also may be a path to having shell access to the database server itself.
Basic Goal:

- Complete existing SQL query and add your own SQL to the command so your code is executed, exposing weakness in the back-end implementation.
Can be performed manually, by entering data into each field on the web page and observing behavior.

Can be automated, and let a tool figure out where the issues are.

OWASP Reference:

https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)
On the "Login" page of the dojo basic:

- 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?
Lab 2 - SQLMap Exercise

- 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

Interactive shell
-sqlmap-shell

OS interaction
-os-shell
Basic options listed

Usage: python sqlmap.py [options]

Options:
- -h, --help            Show basic help message and exit
- -hh                   Show advanced help message and exit
- --version             Show program's version number and exit
- -v, VERBOSITY         Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the target(s)

- -u URL, --url=URL     Target URL (e.g. "http://www.site.com/vuln.php?id=1")
- -g GOOGLEDLORK        Process Google dork results as target URLs

Request:
  These options can be used to specify how to connect to the target URL

- --data=DATA           Data string to be sent through POST
- --cookie=COOKIE       HTTP Cookie header value
- --random-agent        Use randomly selected HTTP User-Agent header value
- --proxy=PROXY          Use a proxy to connect to the target URL
- --tor                 Use Tor anonymity network
- --check-tor           Check to see if Tor is used properly
Try login url:

Lab 2 - SQLMap

Try login url with parameters we used for fuzzing -

sqlmap -u http://dojo-basic/index.php?page=login.php\&user_name=admin\&password=bob

Remember to escape the “&” so the shell doesn’t parse it.

Accept the defaults for any prompts from SQLMap
Try other commands - not all may work!

Maybe a different URL - is there another page with a password field?

Can we inject there?
Lab - SQLMap - other commands

User Info page is also vulnerable

Parameter view_user_info is injectable

URL: http://dojo-basic/index.php?page=user-info.php&view_user_name=admin

Now try other commands: -dbs, -tables, -dump
LAB 3 - PASSIVE RECON
Passive Recon is all about knowing everything there is to know about your target with only public sources available to you, and to not raise any flags that your target would notice.
This includes:
- IP domains and sub-domains
- People
- Infrastructure and Technologies
- Content and logs of potential interest

Tools such as:
- whois
- dns
- shodan
- google-fu
- pastebin (maybe someone else came before you)
Lab 3 - 20 minutes - The Tools

- Shodan
- Whois
- DNS
- Recon-NG

All of these tools are installed on the notebook in front of you.

Start up chrome and go to shodanhq.com, login using the credentials taped to your keyboard (each system has a different set of credentials)

<have them do searching for targets : [list needed]>

Whois : on the command line, go over whois commands and the output
LAB 4 - ACTIVE RECON AND EXPLOTAION (CHEATING)
Active Recon and Exploitation has gone full script-kiddie. In Jan 2018 a new tool called Autosploit was posted on GitHub.

This tool uses Shodan and MSF to automatically do recon and exploit targets.

https://github.com/NullArray/AutoSploit

This is cheating to win, ultimate edition.
From the CLI we will start autosploit.

We will point autosploit to internal classroom server.

We will watch it do its thing.

How does AutoSploit work and how is this cheating to win?
Useful Links

- [http://n0p.net/penguicon/php_app_sec/mirror/xss.html](http://n0p.net/penguicon/php_app_sec/mirror/xss.html) - XSS cheat
- [https://highon.coffee/blog/lfi-cheat-sheet/](https://highon.coffee/blog/lfi-cheat-sheet/) - LFI Cheat
- [https://highon.coffee/blog/reverse-shell-cheat-sheet/](https://highon.coffee/blog/reverse-shell-cheat-sheet/) - Reverse Shell Cheat
- [https://www.gracefulsecurity.com/path-traversal-cheat-sheet-windows/](https://www.gracefulsecurity.com/path-traversal-cheat-sheet-windows/) - Path Traversal Cheat
Docker Images/tools

- docker pull kalilinux/kali-linux-docker official Kali Linux
- docker pull owasp/zap2docker-stable - official OWASP ZAP
- docker pull wpscanteam/wpscan - official WPScan
- docker pull pandrew/metasploイト - docker-metasploイト
- docker pull citizenstig/dvwa - Damn Vulnerable Web Application (DVWA)
- docker pull wpscanteam/vulnerablewordpress - Vulnerable WordPress Installation
- docker pull hmlio/vaas-cve-2014-6271 - Vulnerability as a service: Shellshock
- docker pull hmlio/vaas-cve-2014-0160 - Vulnerability as a service: Heartbleed
- docker pull opendns/security-ninjas - Security Ninjas
- docker pull usertaken/archlinux-pentest- lxde - Arch Linux Penetration Tester
- docker pull diogomonica/docker-bench-security - Docker Bench for Security
- docker pull ismisepaul/securityshepherd - OWASP Security Shepherd
- docker pull danmx/docker-owasp-webgoat - OWASP WebGoat Project docker image
- docker pull citizenstig/nowasp - OWASP Mutilillade II Web Pen-Test Practice Application