Security

Vulnerability Testing

Vulnerabilities

- Vulnerabilities appear everywhere in the stack
  - Modern systems are very large and complex
  - Impossible to test all possible use cases in advance
- Long history of
  - Network protocol vulnerabilities
  - OS vulnerabilities
  - Application vulnerabilities
    - Browsers, web servers, database mgmt systems, mail programs
    - Web apps (see OWASP Top 10)
    - Mobile apps
- Also non-technical vulnerabilities”
  - Social engineering
  - Illness, loss of personnel
  - Power failure, comms problems, fire, flood, earthquake, …

Tracking vulnerabilities

- Repositories
  - CVE: Common Vulnerabilities and Exposures
    - Unique ID assigned to each vulnerability identified, e.g. CVE-2017-7269
    - https://cve.mitre.org/
  - CWE: Common Weakness Enumeration (cwe.mitre.org)
  - CVSS: Common Vulnerability Scoring System
    - For assessing severity of a problem
  - National Vulnerability Database (NVD)
  - SecurityFocus
  - SANS Internet Storm Center
  - CERT (Computer Emergency Response Team)
  - Anti-malware vendors (Symantec, Kaspersky, AVG, etc)

Software Vulnerability Testing

- Find flaws in the code early
- Many different techniques
  - Static (against source or compiled code)
    - Security focused static analysis tools
    - Peer review process
    - Formal security code review
  - Dynamic (against running code)
    - Scanning
    - Penetration testing
- Goal
  - Ensure completeness (across all vulnerability areas)
  - Ensure accuracy (minimize false alarms)

Software Vulnerability Testing

DAST:
Dynamic Application Security Testing (focus on running app)

SAST:
Static Application Security Testing (focus on source code)

Manual Penetration Testing

Manual Code Review

Automated Vulnerability Scanning

Automated Static Code Analysis

Secure Code Review

- Static / dynamic analysis tools available
- Requires manual inspection too
- Benefits:
  - Improves code quality
  - Prevents security bugs
  - Increased developer awareness and understanding

Vulnerability Patterns

```
public class DamagedStrutsForm extends ActionForm
{
  public void doForm( HttpServletRequest request) {
     UserBean u = session.getUserBean();
     u.setName(request.getParameter("name"));
     u.setFavoriteColor(request.getParameter("color"));
  }

  public boolean validate( HttpServletRequest request) {
     try {
       if ( request.getParameter("name").indexOf("<scri") != -1 ) {
         logger.log("Script detected" );
         return false;
       }
     }
     catch( Exception e ) {}
     return true;
  }
}
```

Failure to Validate

Failure to Validate

Blacklist Validation

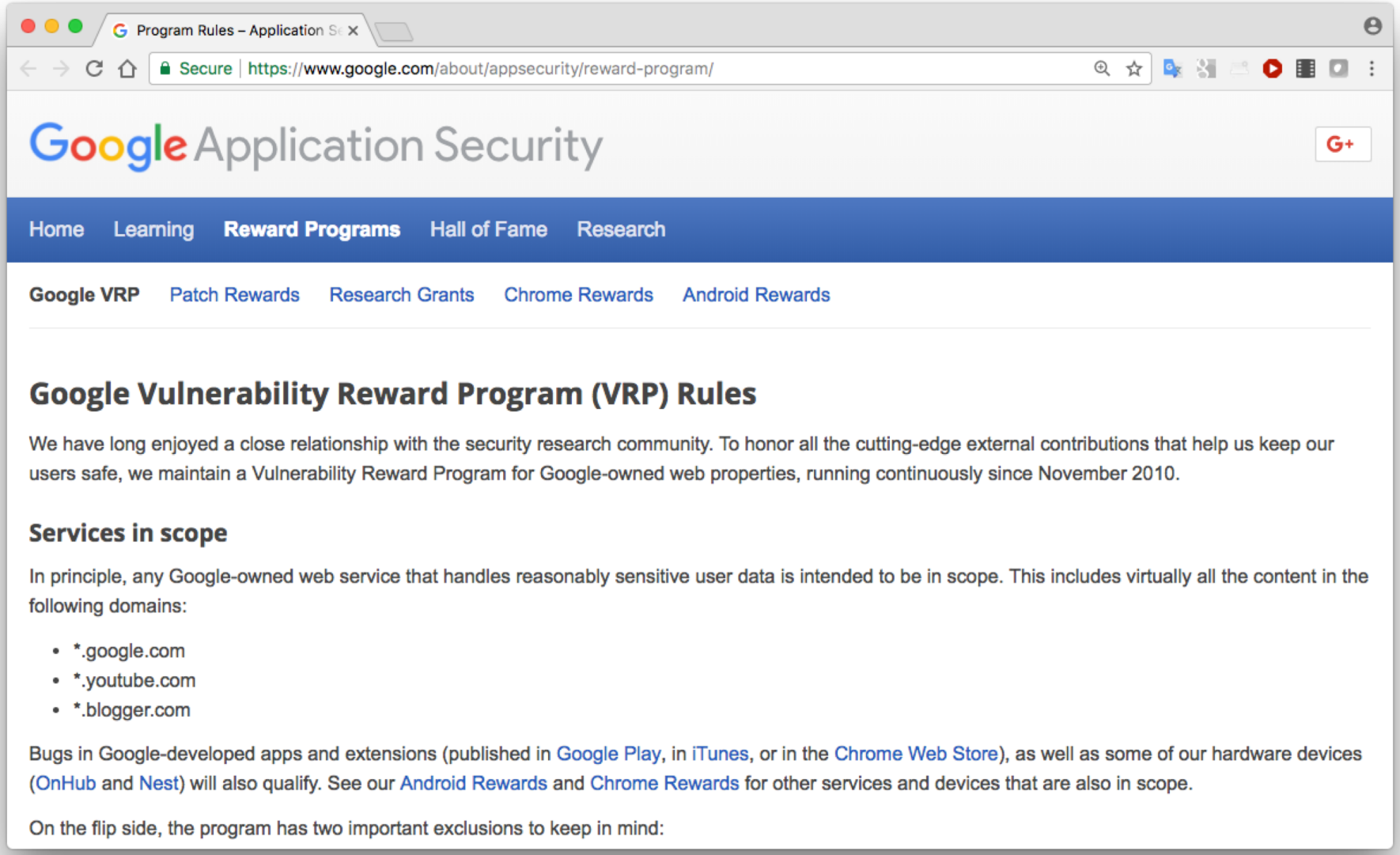
Fail Open

Ethical Hacking

Ethical Hacking...

- Also known as **Penetration Testing**
- Searching for weaknesses and vulnerabilities
- Trying out known attacks
- **Authorised** breaking into systems
  - You MUST have permission from the system's owner!

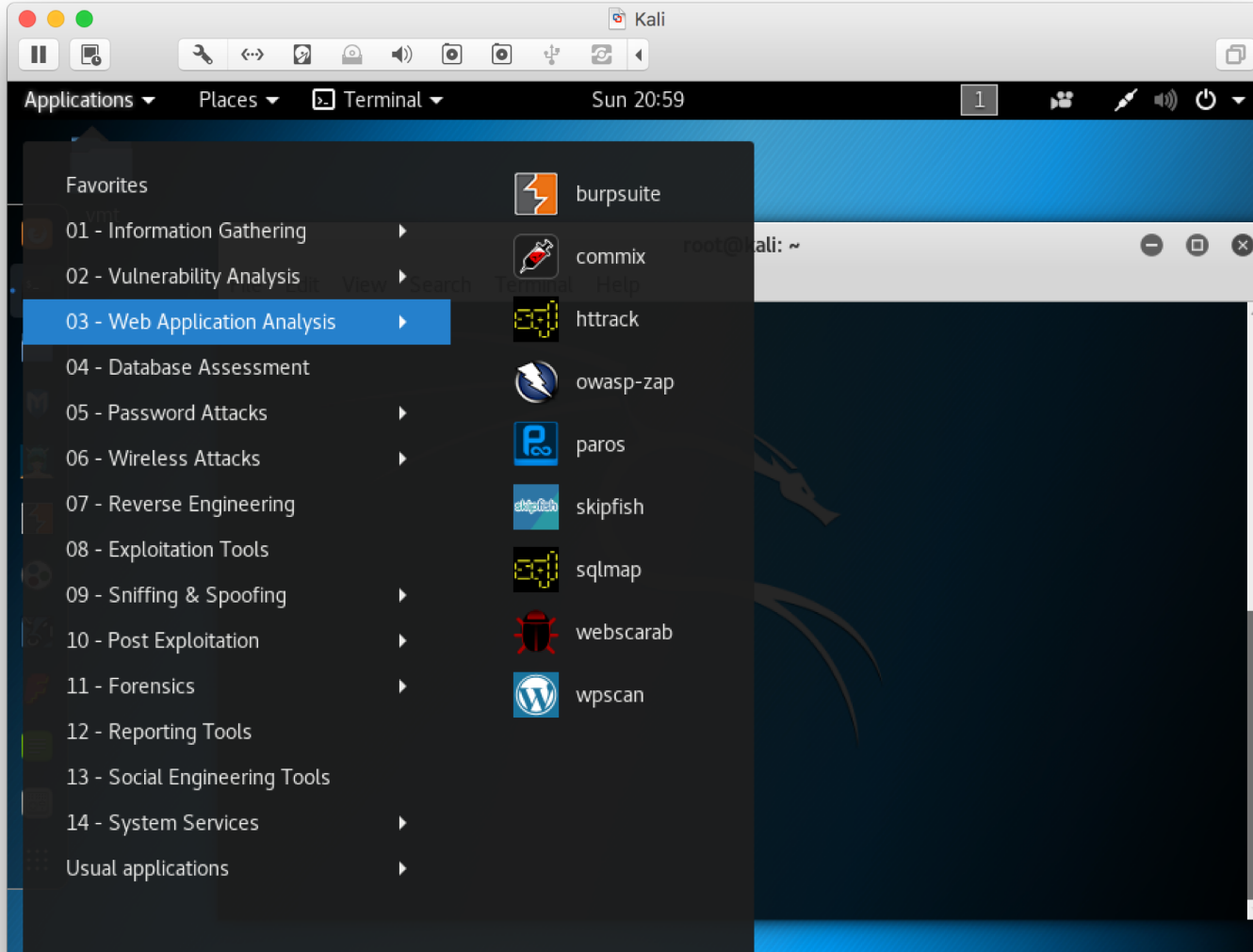
Some companies offer bounties...



KALI LINUX

Advanced penetration testing and security auditing Linux distribution

- 600+ built-in penetration testing tools
- Including web app attack tools
- Free & open source
- Secure environment



Stages of an attack

- Reconnaissance
  - Accessing public information (whois, DNS, web searches, social media posts), “Google hacking”, Maltego, social engineering, …
- Scanning
  - Port scanning (nmap), software version-mapping, automated vulnerability scanning tools, specialist search engines (shodan.io), …
- Exploit systems
  - Authentication grinding (password cracking), passive and active sniffing, buffer overflows, session hijacking, DNS cache poisoning, denial of service, web application attacks, …

Stages of an attack (continued)

- Keeping access
  - Having gone to the trouble of breaking in, the attacker wants to get back in easily, and facilitates this by installing back doors and/or remote control software
  - Trojan horses, netcat listeners, rootkits
- Covering tracks
  - File hiding, log editing, use of covert channels (steganography)

General multi-purpose web app attack tools

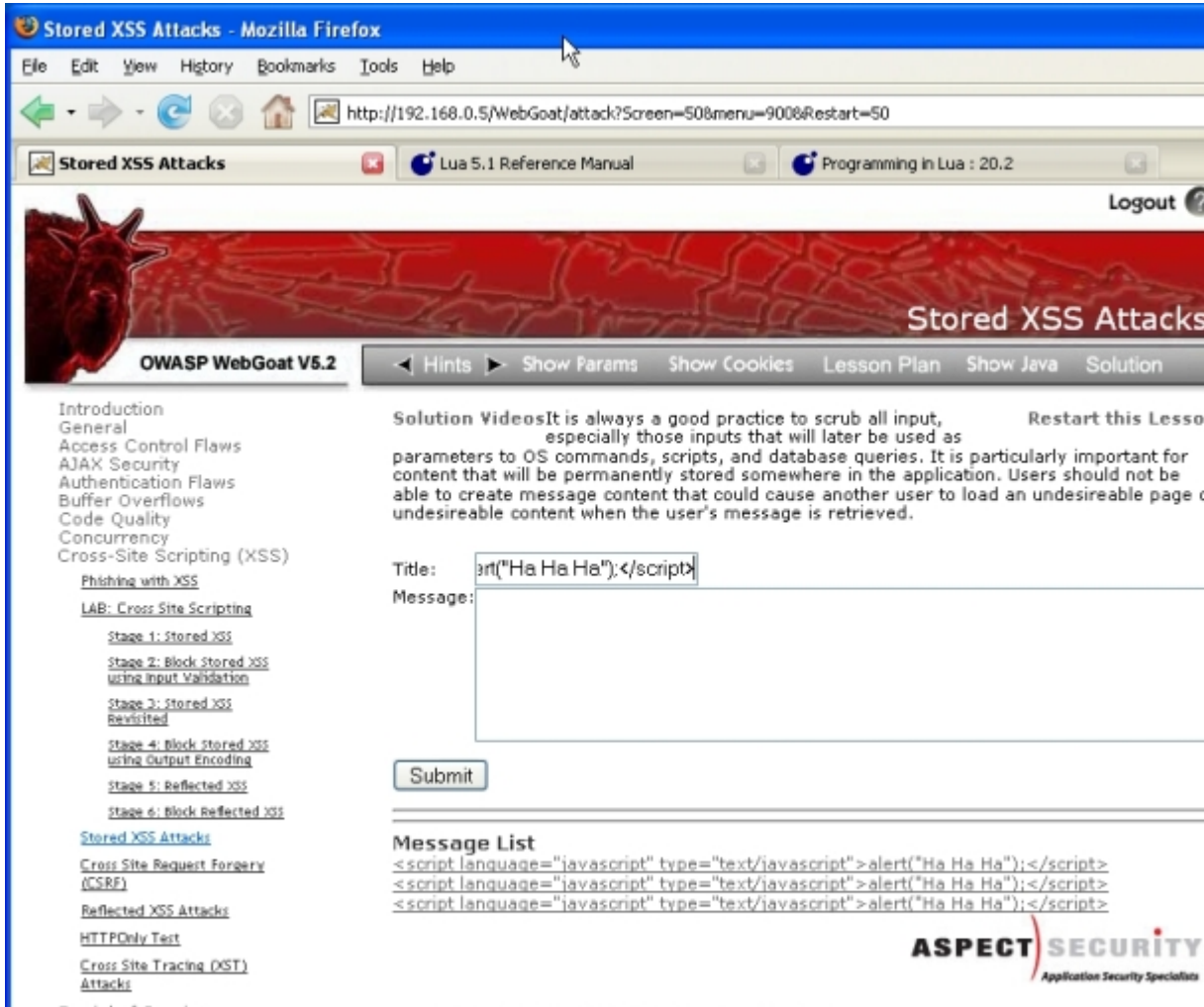
- Typical features
  - Proxy for traffic interception/modification
  - Vulnerability scanning
  - Site crawling
  - Fuzzing
- Popular tools
  - Burp suite
    - Commerical and free/community edition
  - OWASP Zed Attack Proxy (ZAP)
    - Free and open source
  - W3AF
    - Free and open source

Tools for specific purposes

- Sniffing
  - Wireshark
- Port scanning
  - Nmap, netcat
- Proxies for intercepting/modifying traffic
  - WebScarab, Paros proxy, …
- Tools for specific attack types
  - Commix – command injection
  - sqlmap – SQL injection
  - Skipfish – maps site by crawling links and dictionary-based guessing
  - setoolkit – includes site cloning for phishing attacks

Deliberately vulnerable web apps

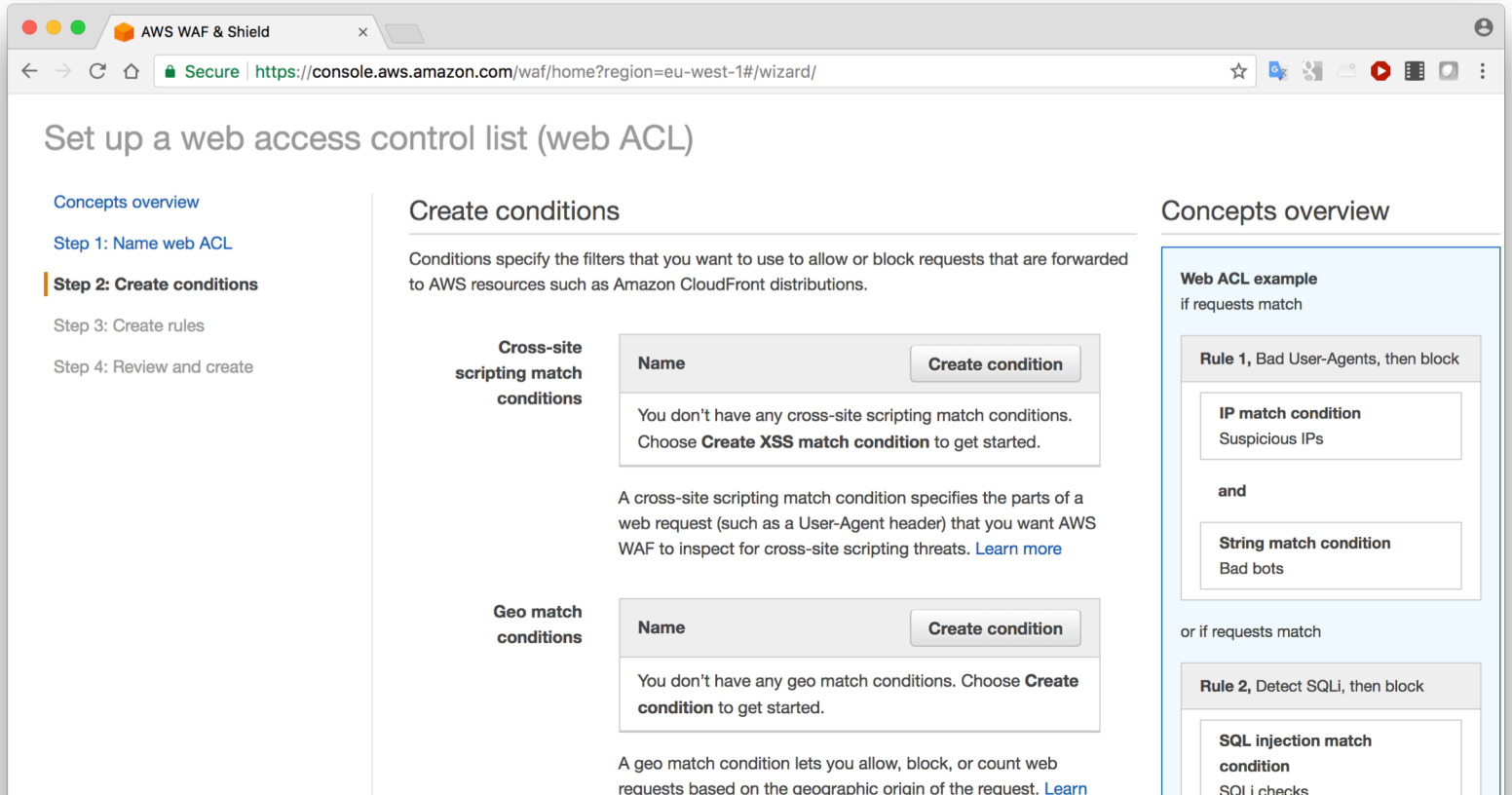
- Good for practicing ethical hacking
- Examples
  - OWASP WebGoat
  - DVWA (Damn Vulnerable Web App)
  - + many others…



Web Application Firewall

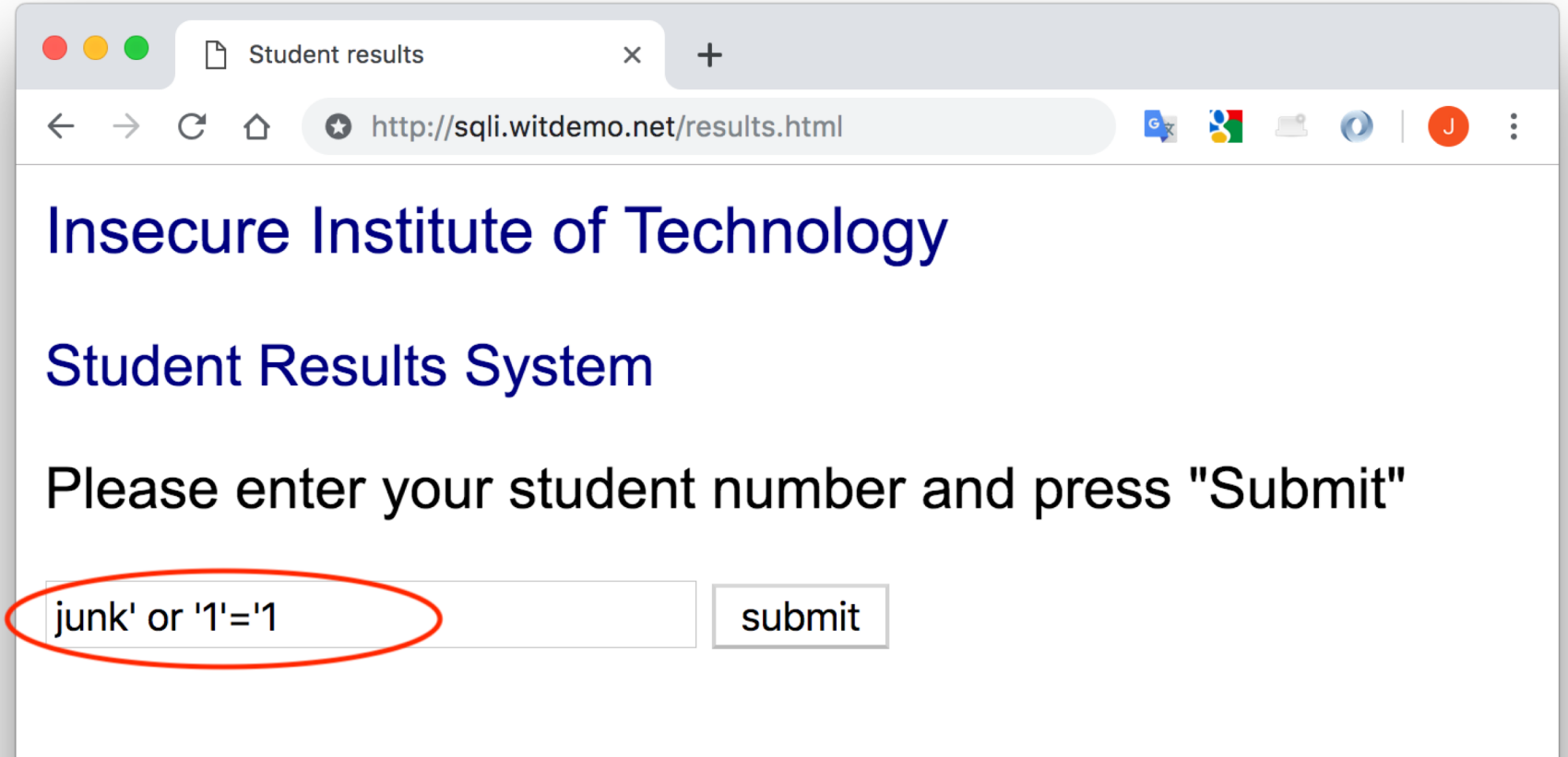
- Protects web applications
- Applies a set of rules to incoming HTTP requests and outgoing responses & logs/monitors/filters accordingly
  - Typically looks for SQLi, XSS,, known vulnerabilities, ...

AWS WAF config



Common attack vector: Malformed input

- Inputting data (e.g. in a web form) to cause a program to behave unusually.
- Often takes advantage of known vulnerability



Common attack vector: Malformed input

- Common types of malformed input attack:
  - SQL injection
  - Buffer overflow
  - Cross site scripting (XSS)
  - XML External Entity (XXE) attack

Background: HTTP request types

HTTP Request Methods

- HTTP is a fairly simple protocol with a small number of methods that define actions to be performed on a specified resource (such as a web page), indentifed by a URL.

HTTP Request Methods

Method	Purpose
GET	Requests data from a specified resource
POST	Submits data to be processed to a specified resource
HEAD	Same as GET but returns only HTTP headers and no document body
PUT	Uploads a representation of the specified URI
DELETE	Deletes the specified resource
OPTIONS	Returns the HTTP methods that the server supports
CONNECT	Converts the request connection to a transparent TCP/IP tunnel

HTTP GET v POST

- GET
  - Designed to retrieve resources (often files) from a server
  - However URI syntax allows a lot of flexibility, so it's easy to use GET to send data as part of URI ("URL encoding")

- POST
  - Designed to send data to a web server.
  - Data provided in the body of the message rather than in the URI
  - More flexible and **more secure**
    - URIs usually cached by browsers, and often bookmarked, shared etc
    - URIs usually logged by proxies and web servers

HTTP GET v POST

- **GET example**

```
GET /path/login?username=jbloggs&password=topsecret
HTTP/1.1
Host: www.site.com
User-Agent: Mozilla/5.0 ...
```

- **POST example**

```
POST /path/login HTTP/1.1
Host: www.site.com
User-Agent: Mozilla/5.0 ...
```

Header

```
username=jbloggs&password=topsecret
```

Body