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)

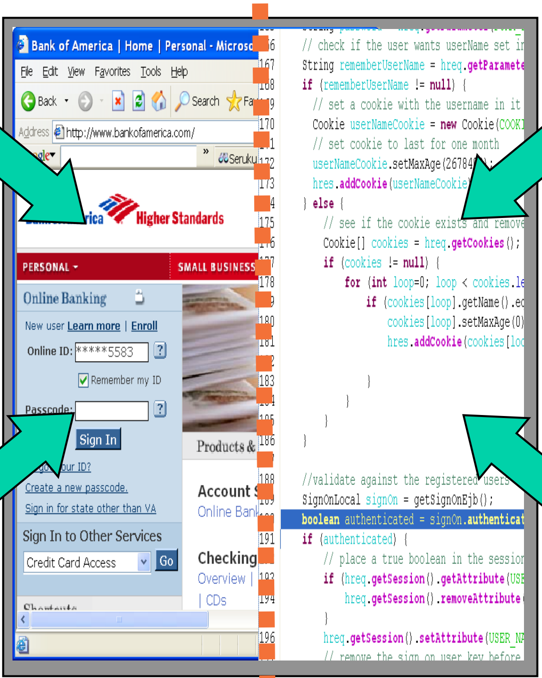
SAST:
Static Application Security Testing
(focus on source code)

Manual Penetration Testing

Manual Code Review

Automated Vulnerability Scanning

Automated Static Code Analysis



```
    // check if the user wants userName set in
    String rememberUserName = hreq.getParamete
    if (rememberUserName != null) {
      // set a cookie with the username in it
      Cookie userNameCookie = new Cookie(COOKI
      // set cookie to last for one month
      userNameCookie.setMaxAge(267840
      hres.addCookie(userNameCookie
    } else {
      // see if the cookie exists and remove
      Cookie[] cookies = hreq.getCookies();
      if (cookies != null) {
        for (int loop=0; loop < cookies.le
          if (cookies[loop].getName().eq
            cookies[loop].setMaxAge(0)
            hres.addCookie(cookies[loo

          }
        }
      }
    }

    //validate against the registered users
    SignOnLocal signOn = getSignOnEjb();
    boolean authenticated = signOn.authenticat
    if (authenticated) {
      // place a true boolean in the session
      if (hreq.getSession().getAttribute(USE
        hreq.getSession().removeAttribute(
      }
      hreq.getSession().setAttribute(USER_NA
      // remove the sign on user key before
```

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

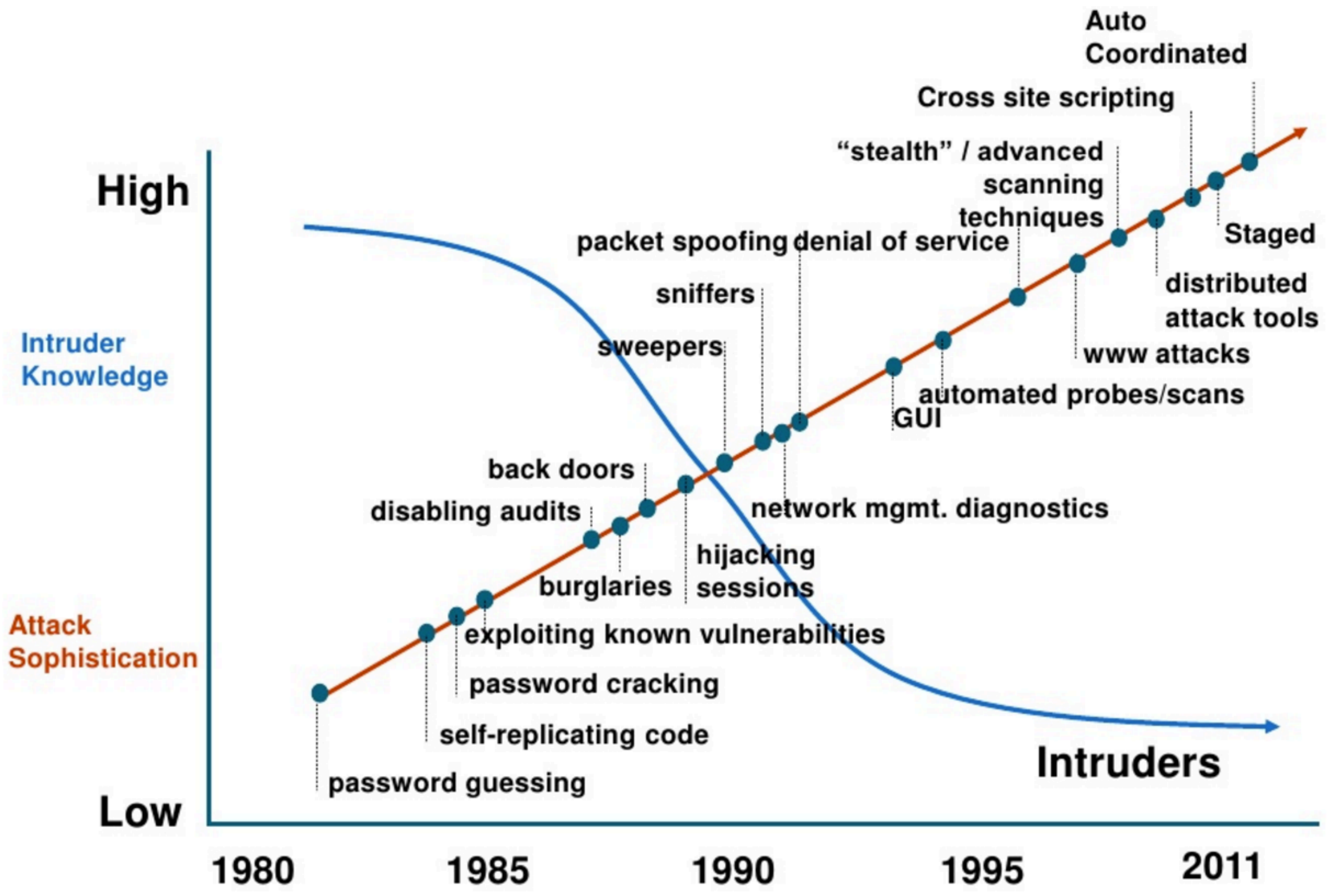
Fail Open

Ethical Hacking

Ethical Hacking...

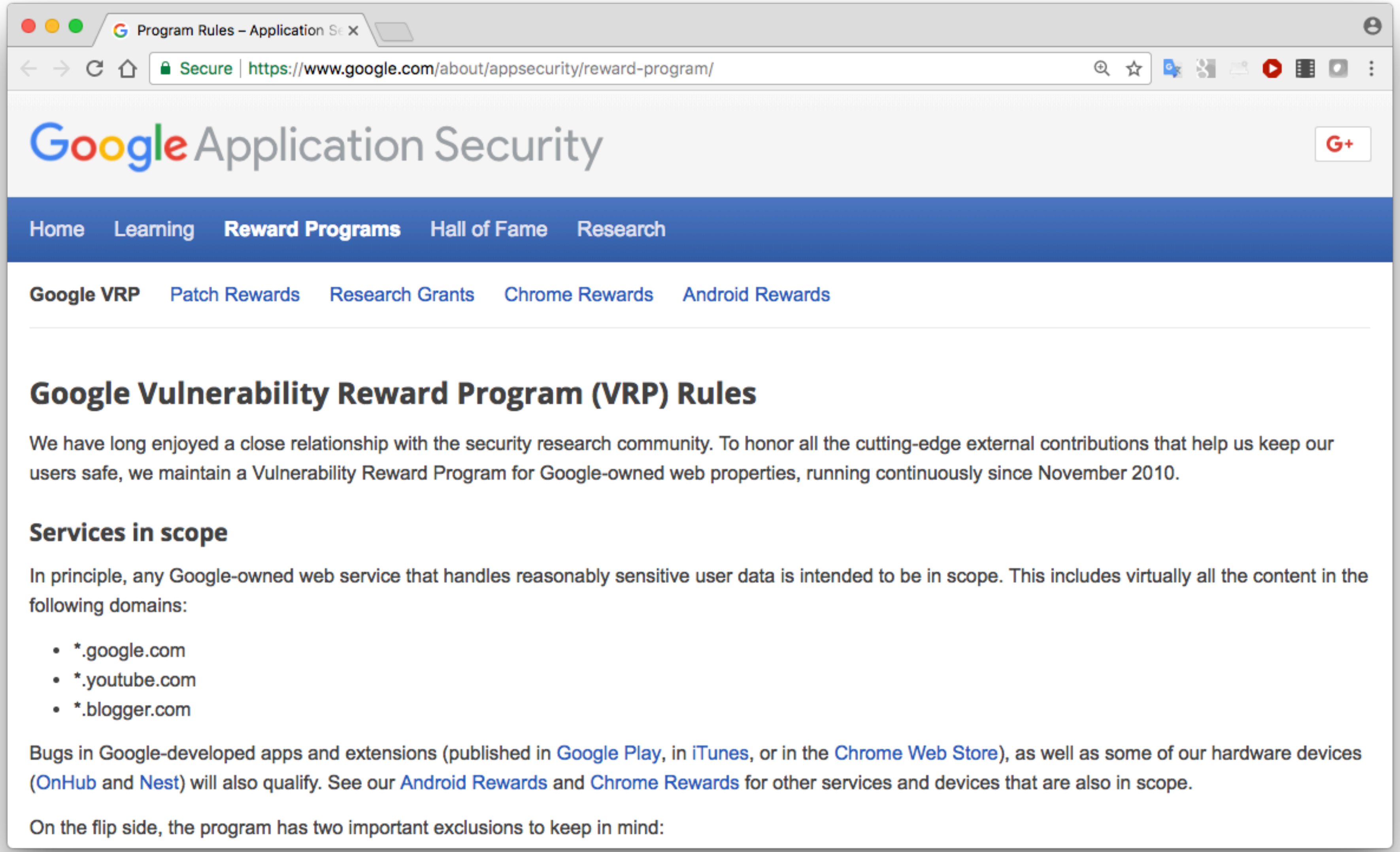
- Also known as **Penetration Testing**
- Searching for weaknesses and vulnerabilities
- Trying out known attacks
- **Authorised** breaking into systems
  - You MUST have permission from the system's owner!

Attack Sophistication vs. Intruder Technical Knowledge



Source: CERT Coordination Center, Pittsburgh

Some companies even offer bounties...


Program Rules – Application Se
Secure https://www.google.com/about/appsecurity/reward-program/
Google Application Security
Home Learning Reward Programs Hall of Fame Research
Google VRP Patch Rewards Research Grants Chrome Rewards Android Rewards

Google Vulnerability Reward Program (VRP) Rules

We have long enjoyed a close relationship with the security research community. To honor all the cutting-edge external contributions that help us keep our users safe, we maintain a Vulnerability Reward Program for Google-owned web properties, running continuously since November 2010.

Services in scope

In principle, any Google-owned web service that handles reasonably sensitive user data is intended to be in scope. This includes virtually all the content in the following domains:

- *.google.com
- *.youtube.com
- *.blogger.com

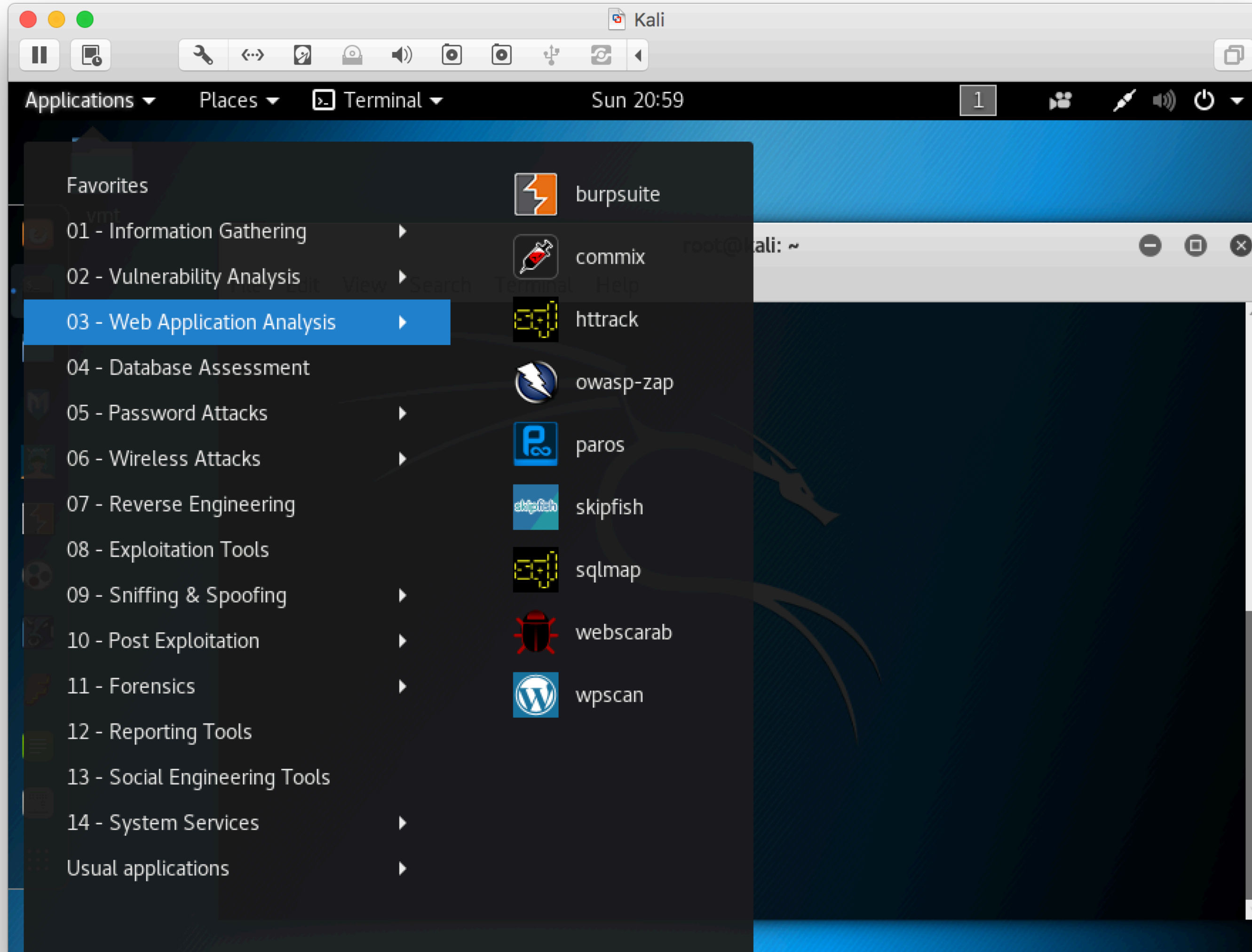
Bugs in Google-developed apps and extensions (published in Google Play, in iTunes, or in the Chrome Web Store), as well as some of our hardware devices (OnHub and Nest) will also qualify. See our Android Rewards and Chrome Rewards for other services and devices that are also in scope.

On the flip side, the program has two important exclusions to keep in mind:

KALI LINUX

Advanced penetration testing and security auditing Linux distribution

- 300+ 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

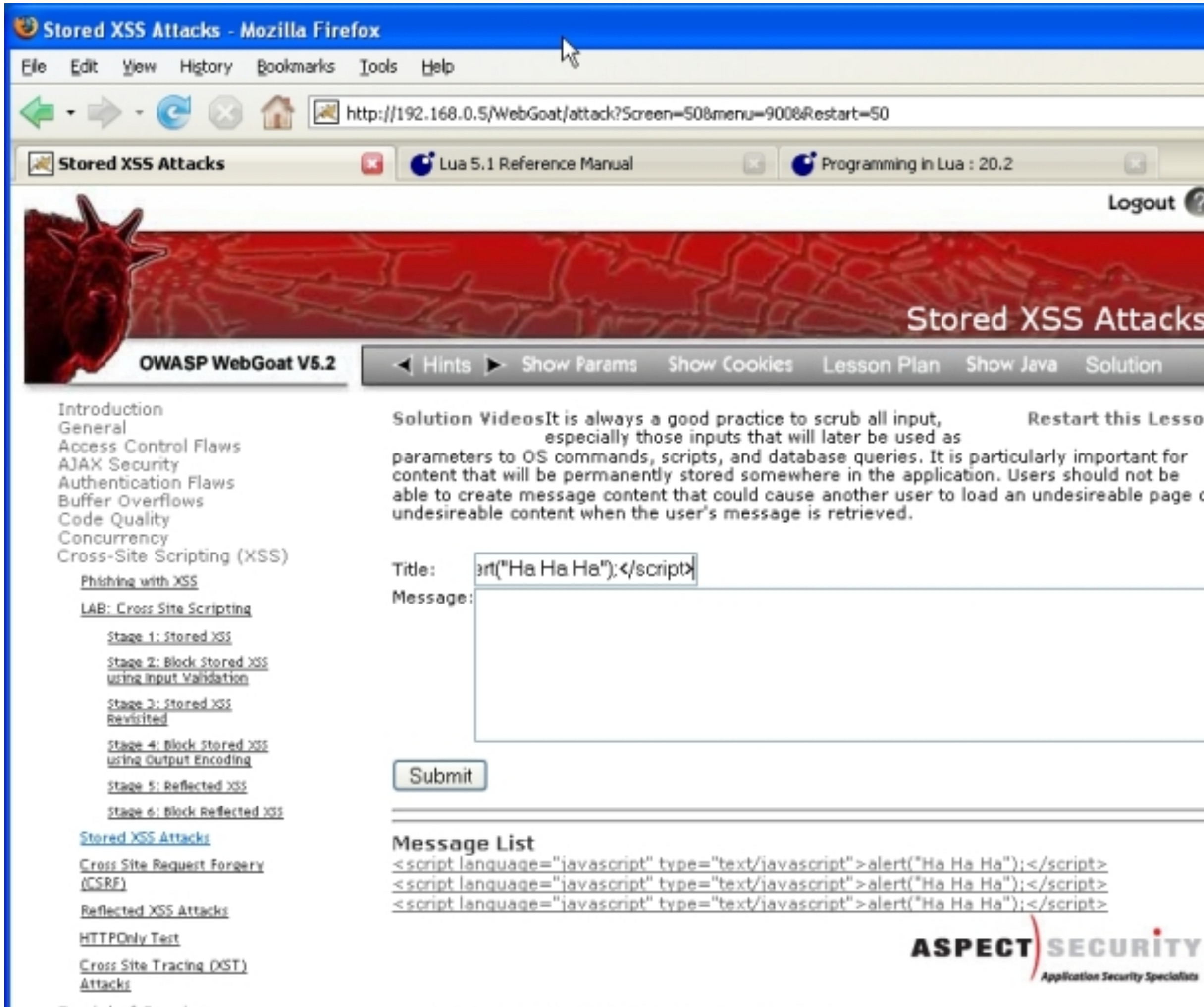
- Typical features
  - Proxy for traffic interception/modification
  - Vulnerability scanning
  - Site crawling
  - Fuzzing
- Popular tools
  - Burp suite
    - Commerical and free/community edition
  - OWASP Zed Attack Proxy (ZAP)
    - Free and open source
  - W3AF

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…


Stored XSS Attacks - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.0.5/WebGoat/attack?Screen=50&menu=900&Restart=50
Stored XSS Attacks
Lua 5.1 Reference Manual
Programming in Lua : 20.2
Logout
Stored XSS Attacks
OWASP WebGoat V5.2
Hints
Show Params
Show Cookies
Lesson Plan
Show Java
Solution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Cross Site Request Forgery (CSRF)
Reflected XSS Attacks
HTTPOnly Test
Cross Site Tracing (XST) Attacks
Solution Videos
Restart this Lesso
Submit
ASPECT SECURITY
Application Security Specialists

It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesireable page o undesireable content when the user's message is retrieved.

Title: rt("Ha Ha Ha");</script>

Message:

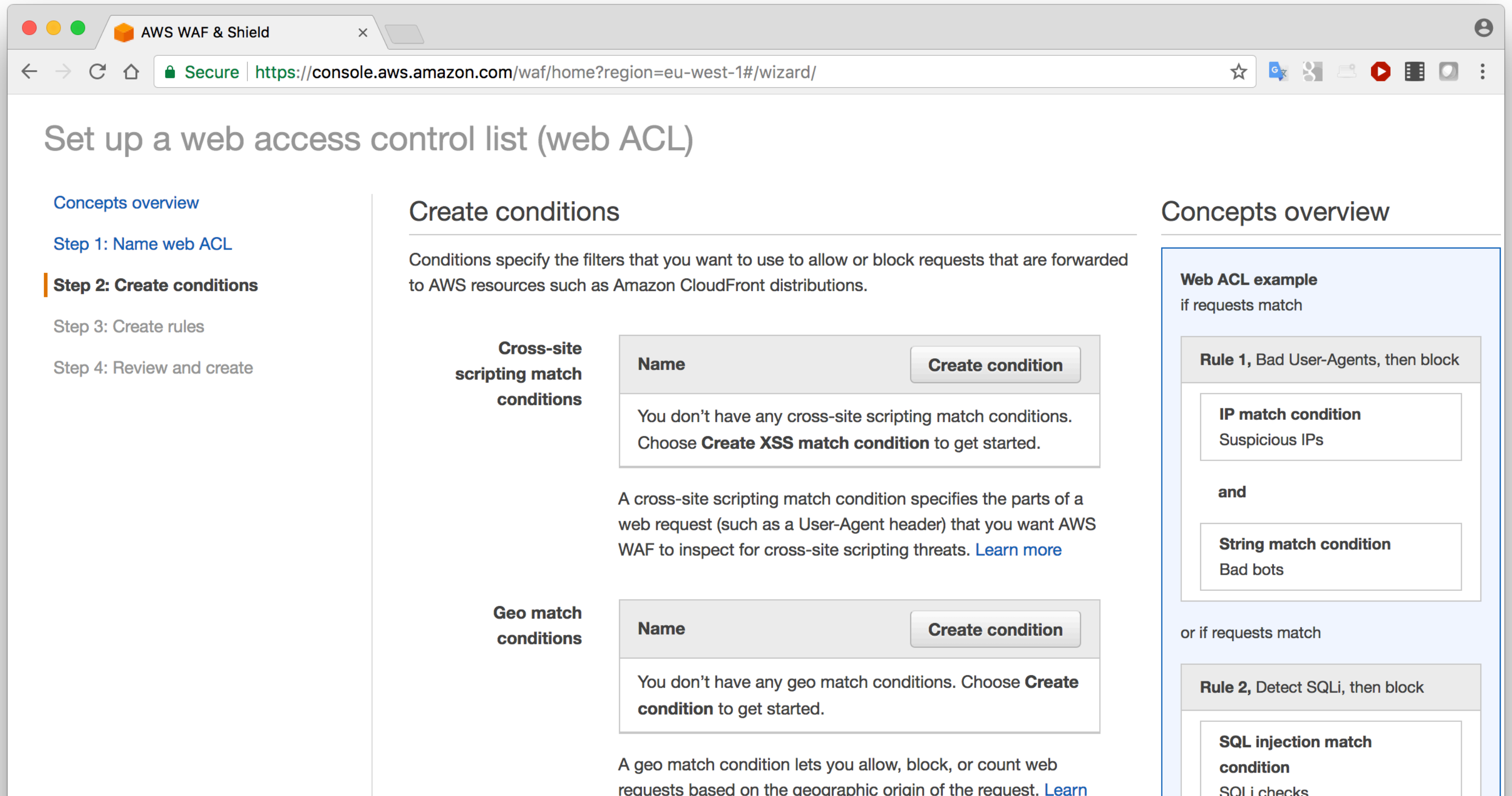
Message List

```
<script language="javascript" type="text/javascript">alert("Ha Ha Ha");</script>
<script language="javascript" type="text/javascript">alert("Ha Ha Ha");</script>
<script language="javascript" type="text/javascript">alert("Ha Ha Ha");</script>
```

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


AWS WAF & Shield
Secure
https://console.aws.amazon.com/waf/home?region=eu-west-1#/wizard/

Set up a web access control list (web ACL)

Concepts overview
Step 1: Name web ACL
Step 2: Create conditions
Step 3: Create rules
Step 4: Review and create

Create conditions

Conditions specify the filters that you want to use to allow or block requests that are forwarded to AWS resources such as Amazon CloudFront distributions.

Cross-site scripting match conditions

Name	Create condition
You don't have any cross-site scripting match conditions. Choose **Create XSS match condition** to get started.	

A cross-site scripting match condition specifies the parts of a web request (such as a User-Agent header) that you want AWS WAF to inspect for cross-site scripting threats. Learn more

Geo match conditions

Name	Create condition
You don't have any geo match conditions. Choose **Create condition** to get started.	

A geo match condition lets you allow, block, or count web requests based on the geographic origin of the request. Learn

Concepts overview

Web ACL example
if requests match

Rule 1, Bad User-Agents, then block

IP match condition
Suspicious IPs

and

String match condition
Bad bots

or if requests match

Rule 2, Detect SQLi, then block

SQL injection match condition
SQLi checks

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

Java

```
void method (String filename) {
   Runtime.getRuntime().exec("more " + filename);   //BAD
   ...
}
---
filename =“xyz.html; /bin/rm -rf /*”;  // malicious argument
```

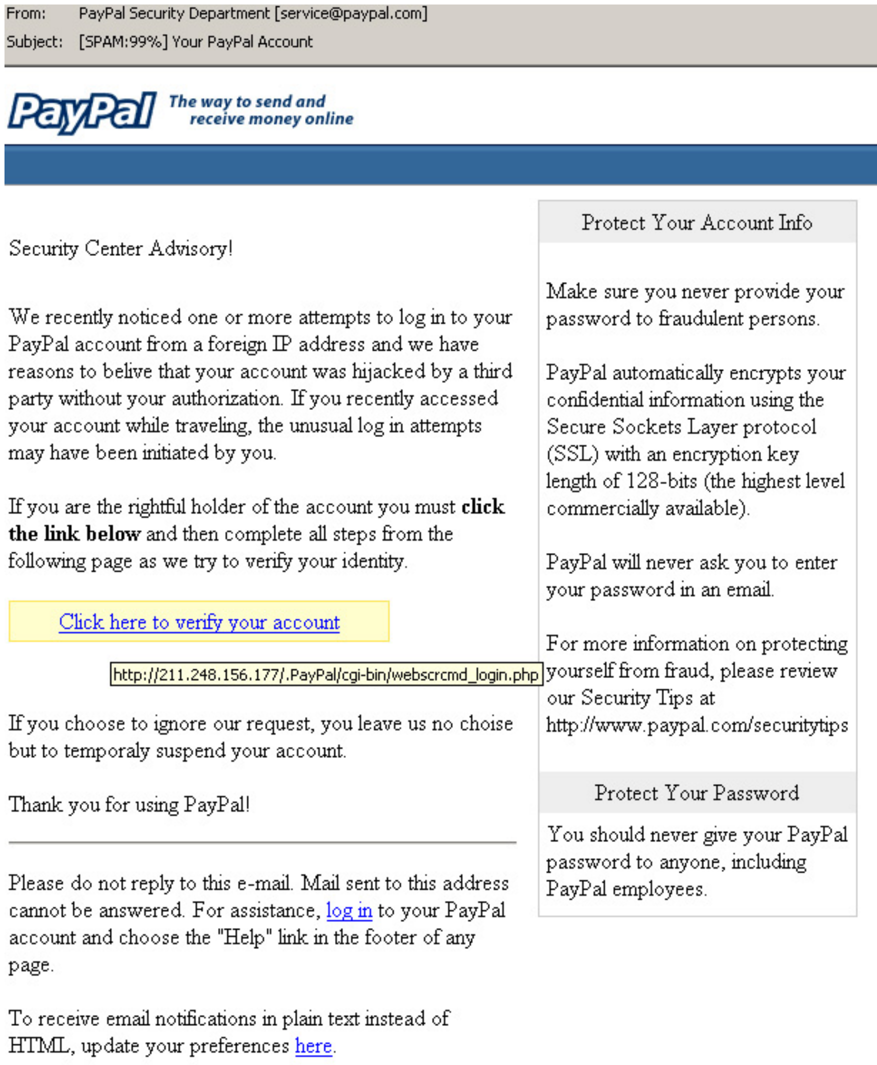
Common attack vector: Malformed input

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

Common attack vector: Phishing

- Forged web pages created to fraudulently acquire sensitive information
- User typically solicited to access phished page from spam email
- Most targeted sites
  - Financial services (banks, etc.)
  - Payment services (e.g., PayPal)
  - Auctions (e.g., eBay)
- Average of over 100,000 unique phishing websites detected <u>per month</u> in 2016
  - Using over 10,000 unique domains (per month)

  [Source: Anti-Phishing Working Group]
- Methods to avoid detection
  - Misspelled URL
  - URL obfuscation
  - Removed or forged address bar



From: PayPal Security Department [service@paypal.com]
Subject: [SPAM:99%] Your PayPal Account

Security Center Advisory!

We recently noticed one or more attempts to log in to your PayPal account from a foreign IP address and we have reasons to belive that your account was hijacked by a third party without your authorization. If you recently accessed your account while traveling, the unusual log in attempts may have been initiated by you.

If you are the rightful holder of the account you must **click the link below** and then complete all steps from the following page as we try to verify your identity.

Click here to verify your account

http://211.248.156.177/.PayPal/cgi-bin/webscrcmd_login.php

If you choose to ignore our request, you leave us no choise but to temporaly suspend your account.

Thank you for using PayPal!

Please do not reply to this e-mail. Mail sent to this address cannot be answered. For assistance, log in to your PayPal account and choose the "Help" link in the footer of any page.

To receive email notifications in plain text instead of HTML, update your preferences here.

Protect Your Account Info

Make sure you never provide your password to fraudulent persons.

PayPal automatically encrypts your confidential information using the Secure Sockets Layer protocol (SSL) with an encryption key length of 128-bits (the highest level commercially available).

PayPal will never ask you to enter your password in an email.

For more information on protecting yourself from fraud, please review our Security Tips at http://www.paypal.com/securitytips

Protect Your Password

You should never give your PayPal password to anyone, including PayPal employees.

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