

Improving Malicious URL Re-Evaluation Scheduling Through an Empirical Study of Malware Download Centers

Kyle Zeeuwen – Sophos, UBC

Matei Ripeanu – University of British Columbia

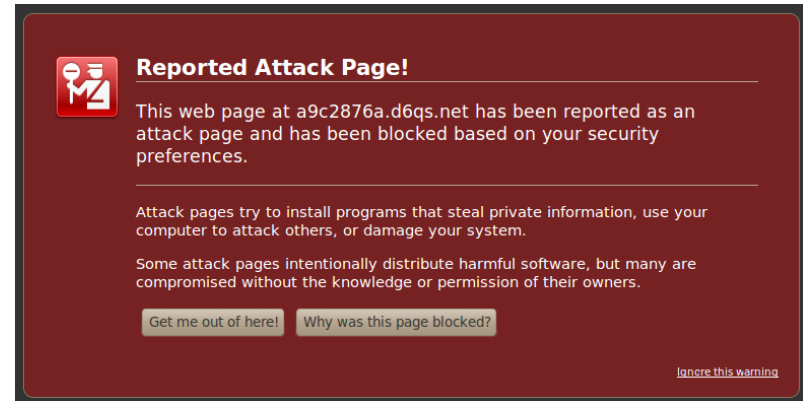
Konstantin Beznosov – University of British Columbia

Outline

- Background and Motivation
- System Design
- Experiment and Results
- Contributions
- Future Work

Why Do Security Researchers Crawl Websites

To determine if a link is malicious

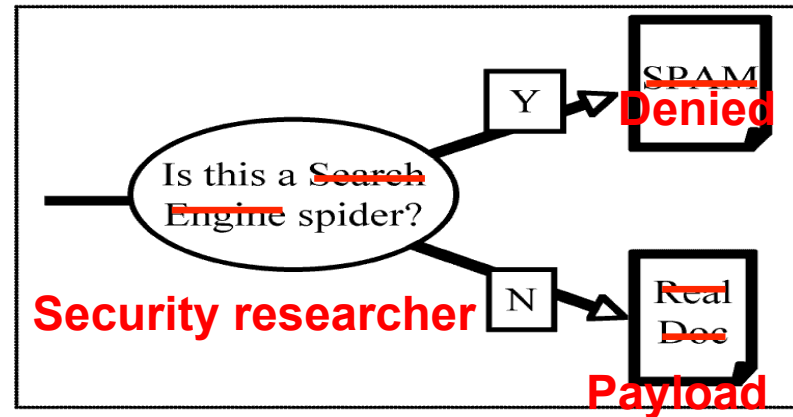


To collect samples and improve detection

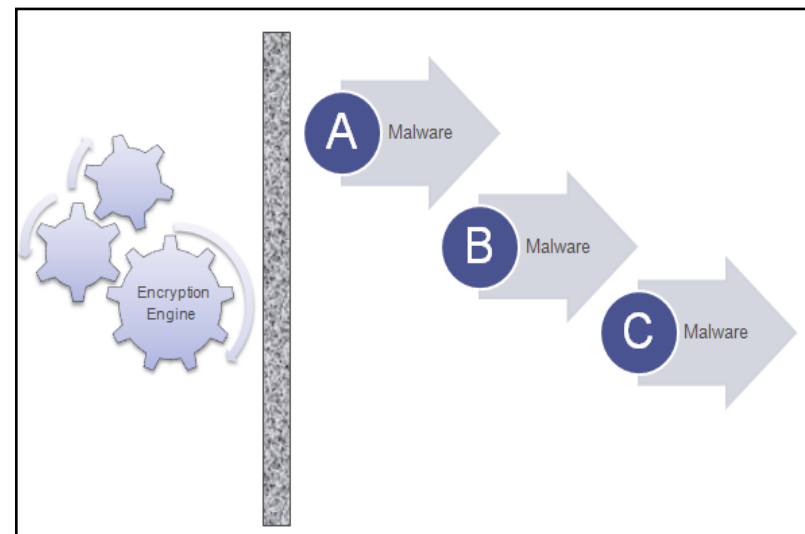


How Malicious Networks Counter Security Researchers

Cloaking



Rapid Sample Updates



System Design

- Objective: Refetch URLs at pre-determined intervals from multiple clients
- Experiment Controller:
 - Select specific URLs, calculate fetch pattern for multiple clients, potentially varying client behaviour
- Fetcher:
 - Low interaction honeyclient
 - Capture DNS, HTTP, downloaded content, and timing info

Experiment Configuration

```
<experiment example>
  experiment_start 2010-08-01 00:00:00
  experiment_end   2010-12-31 23:59:59
  max              100000
  max_per_day     2000
```

```
<criteria>
  url = qr/^[^\./]+\.\ru\/?/
</criteria>
```

```
<downloader researcher>
  client      client1
  <pattern>
    type Tachyon::Experiment::Behavior
    period              3600
    jitter              900
  </pattern>
</downloader>
```

```
<downloader goat>
  client      client2
  <pattern>
    type Tachyon::Experiment::Behavior
    initial_offset  900
    period          129600
    jitter          10800
  </pattern>
</downloader>
</experiment>
```

Accept all .ru URLs



Fetch them:
Every hour \pm 15
minutes

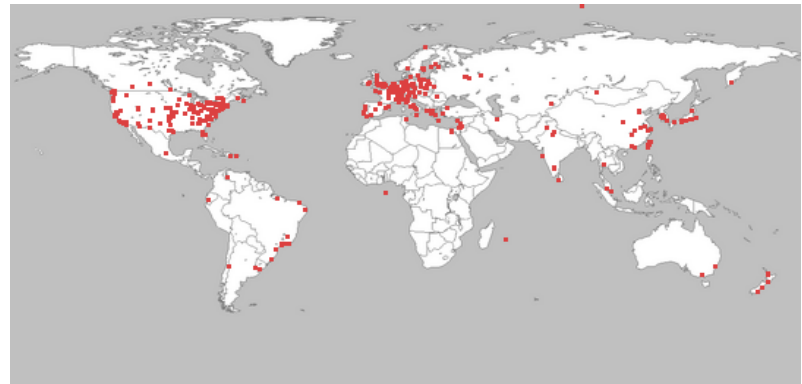


Every 1.5 days \pm 3
hours



Experiments

- Sophos URL feeds of known malicious URLs linking directly to malware
- Aggressively refetch from one IP, slowly refetch from a pool of IPs. Look for differences
 - Planet Lab nodes
 - Linode
 - Personal PCs



Experiments

- 5000 URLs analyzed between Sept 2010 – Feb 2011

1000 - no malware (dead URLs)

2500 - a single sample over time

1500 - multiple samples over time

150 – 200 server side polymorphic URLs*

The rest were 'periodic' updaters

URL count	Number samples
1000	<10
150	+10
60	+25
30	+50
15	+100
20	+200
10	+300

IP Blacklisting Evidence

- 4 domains blacklisted the high volume IPs
- Jan 26 – Feb 2:
 - 1500 requests were answered with 182 distinct malicious samples
 - 2 samples were served 751 times
 - Other samples seen 1-4 times, most often only once
- Feb 2:
 - in a period of 10 minutes all sites start refusing the high volume client
 - Requests to the low volume client are answered for another 24 hours

- Was it the same group?
- WHOIS similarities
 - Same nameservers: ns*.laptopamer.net
 - Same created date, registrar, similar phone #
- URL similarities
 - All .ru domains, 3 end in au.exe
- Infrastructure similarities
 - 105 distinct IPs serving these domains
 - 67 IPs observed on more than one URL
- YES. Same group!

Other IP based responses

Links to 'Swizzor' malware

- 3 different "IP/path.int" URLs fetched by two clients for over a week (over 2000 fetches)
 - All requests from client 1 answered with Swizzor sample "A"*
 - All requests from client 2 answered with Swizzor sample "B"
- 4th URL same behavior but with two new samples "C" and "D"

* At two instances new samples were served ("E","F"), but immediately back to sample "A"

Update Frequency

Crawler Objectives

- Don't miss updates to the content
- Conserve Resources
- Get accurate content

cost of missed
samples

success = reduction in fetches - $\alpha \times$ missed samples



Re-calculating the Refetch Interval

- Basics: Assume we fetch a URL twice:
 - If we get a new sample we win*.
 - If we get the same sample we lose*

- Evaluate these functions:

Linear correction function $I(n) = I(n-1) + X$

Percentage correction function $I(n) = I(n-1) \times X$

Exponent correction function $I(n) = I(n-1)^X$

$I(n)$ – the interval to wait before issuing the n^{th} request
 X – a coefficient to vary (win – negative, lose – positive)

Simulated algorithm performance over a range of α

Missed sample cost (α)	Fetch Reduction (%)	Missed Samples (%)	Correction Function	+X	-X
0-2	94%	56%	exponent	0.12	0.03
3	93%	49%	exponent	0.09	0.03
4-6	90%	40%	exponent	0.06	0.03
7-8	86%	32%	percentage	0.3	0.15
9-10	85%	30%	percentage	0.3	0.2
11-12	83%	28%	percentage	0.3	0.25
13-19	82%	26%	percentage	0.3	0.3
20-21	78%	24%	percentage	0.2	0.25
22-33	76%	22%	percentage	0.15	0.2
34-57	68%	19%	percentage	0.1	0.2
58-62	59%	17%	percentage	0.05	0.1
63-9000	36%	12%	linear	300	3300

Results and Contributions

- Provide strong evidence of blacklisting by malware download centers
- Show that a lot of malware can be downloaded for a long time using a simple low interaction honeypoint
- Propose and evaluate strategies for optimizing scarce fetch resources

Future Work

- Evaluate and improve the proposed algorithms
- Compute and incorporate 'missed sample' cost into algorithms
- More investigation of blacklisting

Acknowledgements

- SophosLabs
- PlanetLabs
- University of British Columbia

Thanks!

?

Extra Slides

Other IP based responses

Client1
Client2

http://66.220.17.153/bins/int/9kgen_up.int fetched by lbytebetter: ()
8 fetches from 2011-01-24 13:00:00 to 2011-02-03 22:59:09: HTTP rc: 200 content: df5d0b0e34cb4fde21932b37268ba951005d2ab5 'EXE' **C**
http://66.220.17.153/bins/int/9kgen_up.int fetched by client1: ()
474 fetches from 2011-01-24 12:46:05 to 2011-02-05 02:15:22: HTTP rc: 200 content: 8c480f4e9510622d9c3257877382abc32d10c407 'EXE' **D**

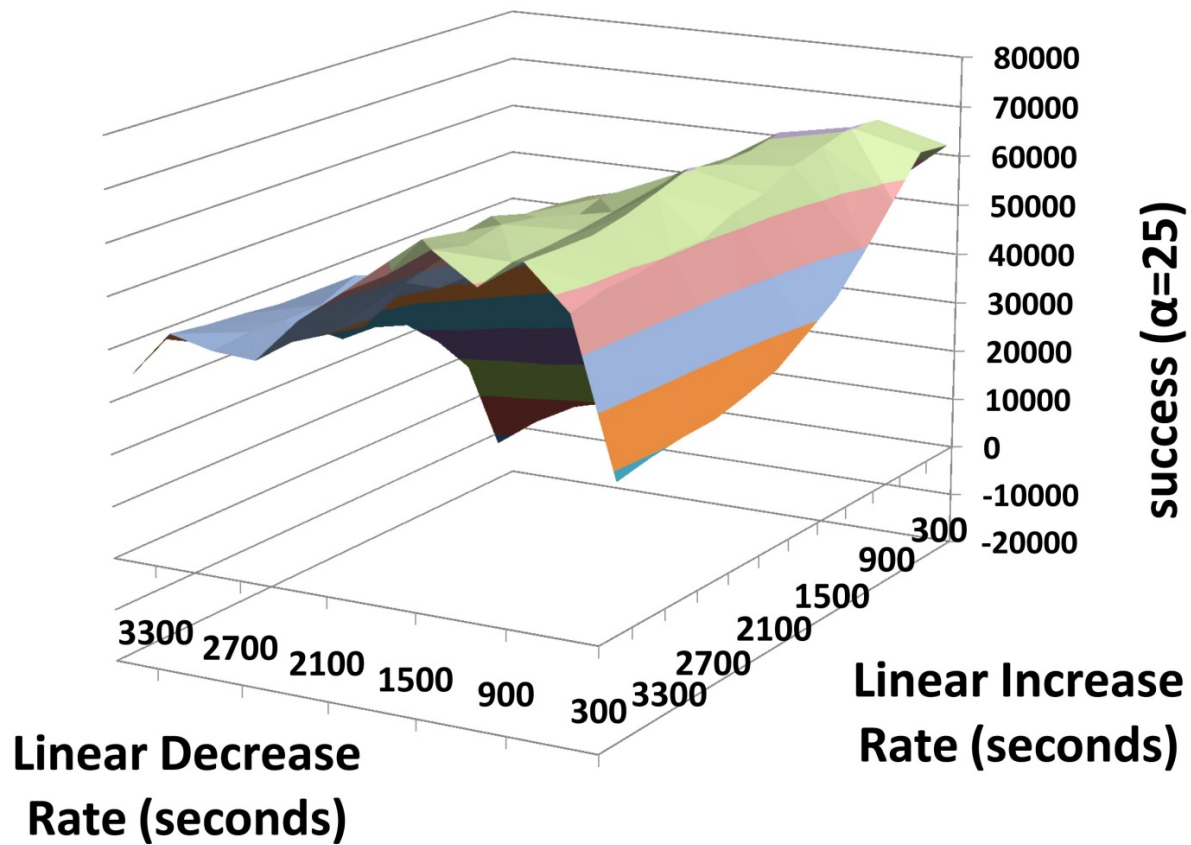
http://66.220.17.157/bins/int/upd_admn.int fetched by lbytebetter: ()
8 fetches from 2011-01-25 03:09:00 to 2011-02-04 10:27:28: HTTP rc: 200 content: 052dd290cc631962f30fa71919fed5f5371802f3 'EXE' **A**
http://66.220.17.157/bins/int/upd_admn.int fetched by client1: ()
246 fetches from 2011-01-25 00:53:28 to 2011-01-30 03:25:54: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**
1 fetches from 2011-01-30 03:52:57 to 2011-01-30 03:52:57: HTTP rc: 200 content: 0fab78c1287e68032c2add1b71f55294ec65028 'EXE' **E**
204 fetches from 2011-01-30 04:23:26 to 2011-02-05 02:21:49: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**

http://66.220.17.156/bins/int/upd_admn.int fetched by lbytebetter: ()
8 fetches from 2011-01-25 02:51:18 to 2011-02-04 11:06:29: HTTP rc: 200 content: 052dd290cc631962f30fa71919fed5f5371802f3 'EXE' **A**
http://66.220.17.156/bins/int/upd_admn.int fetched by client1: ()
448 fetches from 2011-01-25 01:57:53 to 2011-02-05 02:27:25: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**

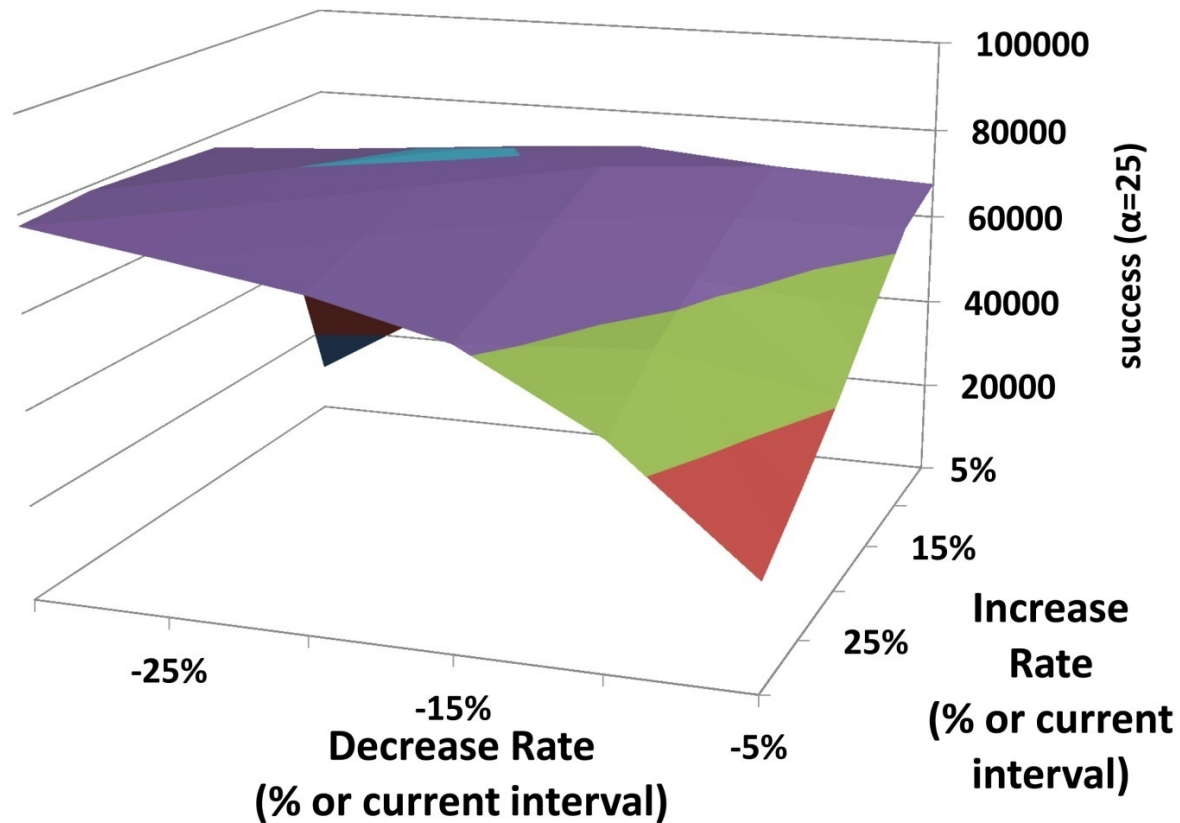
http://66.220.17.155/bins/int/upd_admn.int fetched by lbytebetter: ()
8 fetches from 2011-01-25 02:01:03 to 2011-02-04 13:00:36: HTTP rc: 200 content: 052dd290cc631962f30fa71919fed5f5371802f3 'EXE' **A**
http://66.220.17.155/bins/int/upd_admn.int fetched by client1: ()
449 fetches from 2011-01-25 02:00:22 to 2011-02-05 02:29:41: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**

http://66.220.17.158/bins/int/upd_admn.int fetched by lbytebetter: ()
8 fetches from 2011-01-25 04:46:46 to 2011-02-04 19:15:56: HTTP rc: 200 content: 052dd290cc631962f30fa71919fed5f5371802f3 'EXE' **A**
http://66.220.17.158/bins/int/upd_admn.int fetched by client1: ()
201 fetches from 2011-01-25 04:49:30 to 2011-01-29 08:51:04: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**
1 fetches from 2011-01-29 09:24:01 to 2011-01-29 09:24:01: HTTP rc: 200 content: e6c28ec08af23f0ec3fda30709b434b6bdb25344 'EXE' **F**
241 fetches from 2011-01-29 09:46:12 to 2011-02-05 02:18:32: HTTP rc: 200 content: 987724abab42b13d4c690e5e73dad0cece9c9f61 'EXE' **B**

Linear Correction Function Results



Percentage Correction Function Results



Exponent Correction Function Results

