Northeastern University
Khoury College of Computer Sciences

iseclab

NEU SecLab

Understanding and Mitigating the Security Risks of Content Inclusion in Web Browsers

PhD Thesis Defense

Sajjad Arshad

Friday, 12 April 2019

Content Inclusion on the Web

- Websites include various types of content to create interactive user interfaces
  - JavaScript
  - Cascading Style Sheets (CSS)
- 93% of the most popular websites include JavaScript from external sources
  - JavaScript libraries are hosted on fast content delivery networks (CDNs)
  - Integration with advertising networks, analytics frameworks, and social media
- Browser extensions enhance browsers with additional capabilities
  - Fine-grained filtering of content
  - Access cross-domain content
  - Perform network requests

Security Risks

- ***Malvertising*** by third-party content
  - Launch drive-by downloads
  - Redirect visitors to phishing sites
  - Generate fraudulent clicks on ads

NEU SecLab

Malvertising

ZDNet VIDEOS 5G WINDOWS 10 CLOUD AI INNOVATION SECURITY MORE NEWSLETTERS ALL WRITERS

MUST READ: Chromium-based Edge: Hands on with Microsoft's new browser

Malvertising campaign targets Apple users with malicious code hidden in images

New malvertising group named VeryMal hijacked over five million web sessions to redirect Apple users to sites offering malware-laced software.

By Catalin Cimpanu for Zero Day | January 24, 2019 -- 15:20 GMT (07:20 PST) | Topic: Security

ZDNet VIDEOS 5G WINDOWS 10 CLOUD AI INNOVATION SECURITY MORE NEWSLETTERS ALL WRITERS

MUST READ: Chromium-based Edge: Hands on with Microsoft's new browser

Double trouble: Two-pronged cyber attack infects victims with data-stealing trojan malware and ransomware

A 'prolific' malvertising campaign has been used to distribute the Vidar information stealer and GandCrab ransomware.

By Danny Palmer | January 7, 2019 -- 15:53 GMT (07:53 PST) | Topic: Security

Security Risks

NEU SecLab

- ***Malvertising*** by third-party content
  - Launch drive-by downloads
  - Redirect visitors to phishing sites
  - Generate fraudulent clicks on ads
- ***Ad(vertisement) injection*** by browser extensions
  - Divert revenue from content publishers
  - Harm the reputation of the publisher from the user's perspective
  - Expose users to malware and phishing

Ad(vertisement) Injection

NEU SecLab

ZDNet | VIDEOS | 5G | WINDOWS 10 | CLOUD | AI | INNOVATION | SECURITY | MORE | NEWSLETTERS | ALL WRITERS

MUST READ: Chromium-based Edge: Hands on with Microsoft's new browser

Chrome extension with millions of users is now serving popup ads

Good extension turns bad and is now showing unwanted ads for an ad-blocker to millions of users.

By Catalin Cimpanu for Zero Day | February 7, 2019 -- 15:57 GMT (07:57 PST) | Topic: Security

ZDNet | VIDEOS | 5G | WINDOWS 10 | CLOUD | AI | INNOVATION | SECURITY | MORE | NEWSLETTERS | ALL WRITERS

MUST READ: Chromium-based Edge: Hands on with Microsoft's new browser

Firms buy popular Chrome extensions to inject malware, ads

Are adware companies offering lucrative deals to acquire popular Chrome extensions -- and the trust of an extension's users?

By Charlie Osborne for Between the Lines | January 20, 2014 -- 11:19 GMT (03:19 PST) | Topic: Google

Security Risks

NEU SecLab

- ***Malvertising*** by third-party content
  - Launch drive-by downloads
  - Redirect visitors to phishing sites
  - Generate fraudulent clicks on ads
- ***Ad(vertisement) injection*** by browser extensions
  - Divert revenue from content publishers
  - Harm the reputation of the publisher from the user's perspective
  - Expose users to malware and phishing
- ***Style injection*** by relative path overwrite (RPO)
  - Sniffing users' browsing histories
  - Exfiltrate secrets from the website

Thesis Contributions

In this thesis, I investigate the feasibility and effectiveness of novel approaches to understand and mitigate the security risks of content inclusion for website publishers as well as their users. I show that our novel techniques are complementary to the existing defenses.

- ➔ Detection of Malicious Third-Party Content Inclusions ⇒ ***Excision***
- ➔ Identifying Ad Injection in Browser Extensions ⇒ ***OriginTracer***
- ➔ Analysis of Style Injection by Relative Path Overwrite (RPO)

Excision

Detection of Malicious Third-Party Content Inclusions

NEU SecLab

NEU SecLab

Third-Party Content Defenses

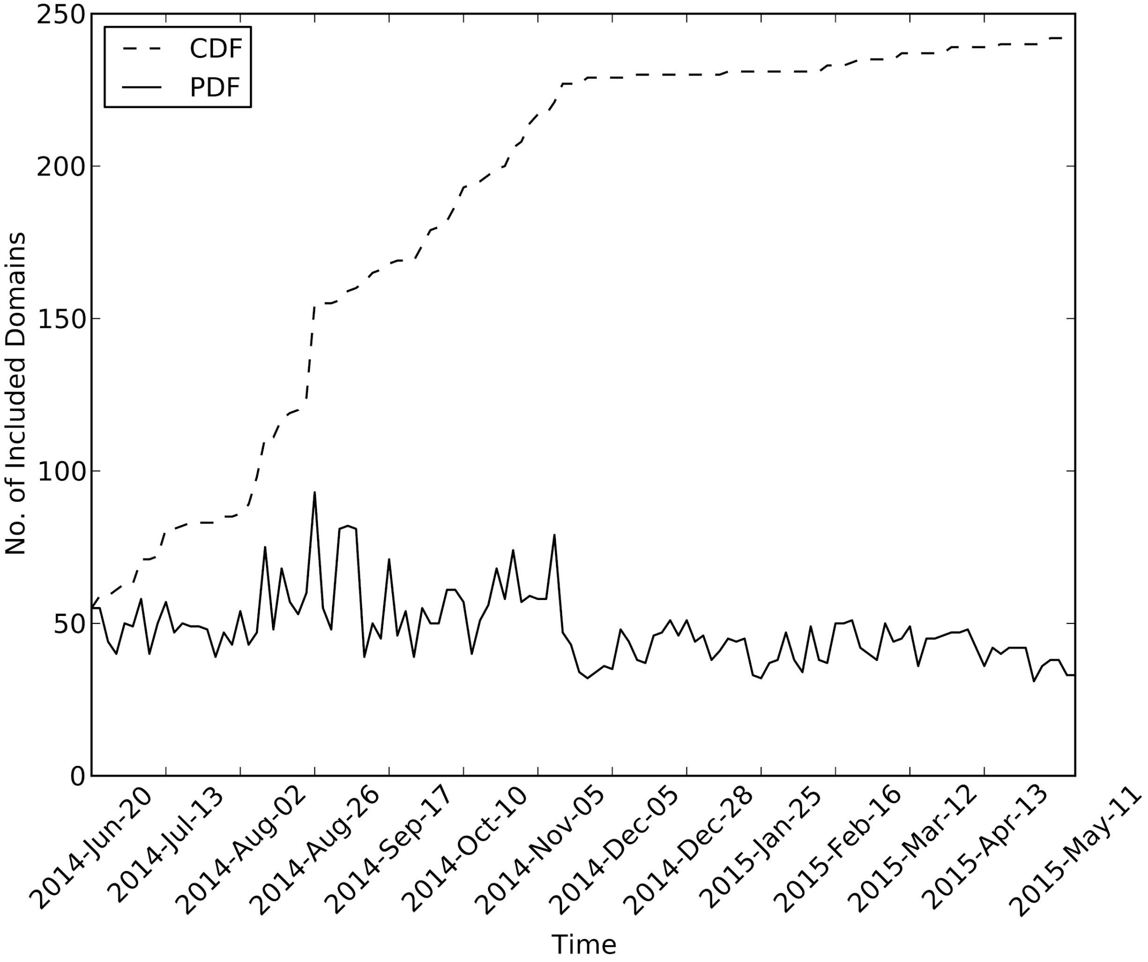
- ➔ Same-origin policy (SOP)
- ➔ iframe-based isolation
- ➔ Language-based isolation
- ➔ Policy enforcement
- ➔ Content Security Policy (CSP)

NEU SecLab

Content Security Policy

Content-Security-Policy: default-src 'self'; script-src: js.trusted.com

- Access control policy sent by web apps, enforced by browsers
- Whitelist of allowed origins to load resource from
- ISPs and browser extensions modify CSP rules
- Non-trivial to deploy
  - Ad syndication or real-time ad auctions
  - Arbitrary third-party resource inclusion by Browser extensions



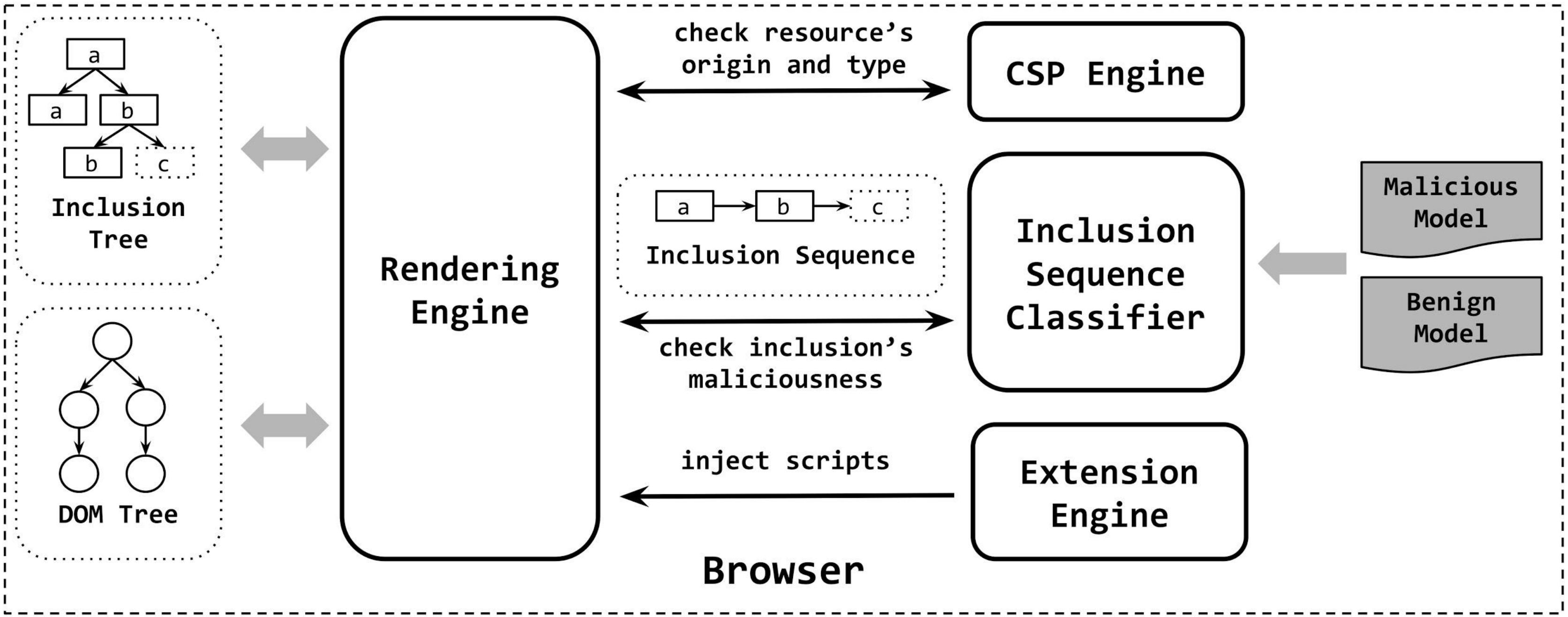
CDF
PDF
No. of Included Domains
Time
0
50
100
150
200
250
2014-Jun-20
2014-Jul-13
2014-Aug-02
2014-Aug-26
2014-Sep-17
2014-Oct-10
2014-Nov-05
2014-Dec-05
2014-Dec-28
2015-Jan-25
2015-Feb-16
2015-Mar-12
2015-Apr-13
2015-May-11

NEU SecLab

Excision

NEU SecLab

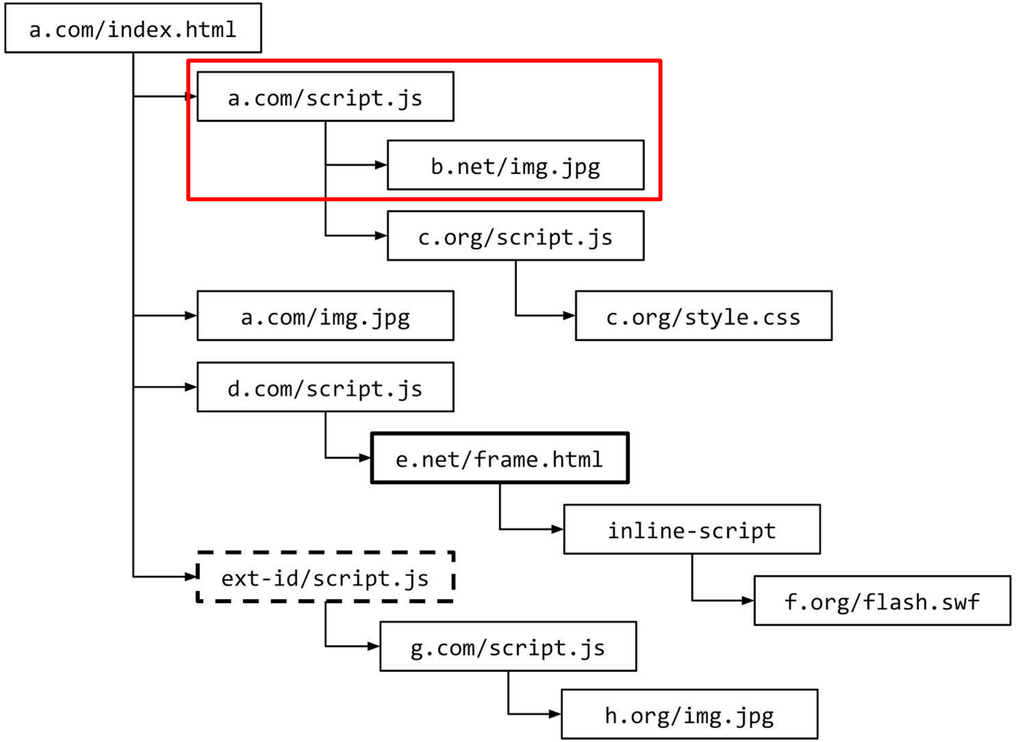
Block malicious content automatically before it attacks the browser



Inclusion Tree

Web Page:**a.com/index.html**

```
<html>
  <head><title>...</title></head>
  <body>
    <ul><li>...</li></ul>
    <a href="…"></a>
    <div>
      <script src="script.js"></script>
      <img src="b.net/img.jpg">
      <script src="c.org/script.js"></script>
      <link href="c.org/style.css">
    </div>
    <img src="img.jpg"/>
    <script src="d.com/script.js"></script>
    <iframe src="e.net/frame.html">
      <html>
        <head></head>
        <body>
          <script>...</script>
          <object data="f.org/flash.swf"></object>
        </body>
      </html>
    </iframe>
    <script src="g.com/script.js"></script>
    <img src="h.org/img.jpg"/>
  </body>
</html>
```



Inclusion Sequence Classification

Goal: Given trained models, label inclusion sequences as either benign or malicious

- ➔ Hidden Markov Models (HMM)
  - ◆ Model inter-dependencies between resources in the sequence
  - ◆ Trained one HMM for the benign class and one for the malicious class
  - ◆ 20 states that are fully connected
- ➔ Training HMM is computationally expensive, but computing the likelihood of a sequence is instead very efficient
  - ◆ Good choice for real-time classification

Classification Features

R0 → R1 → … → Rn ⇒ [F0, …, F24] → [F0, …, F24] → … → [F0, …, F24]

- ➔ Convert the inclusion sequence into sequence of feature vectors
- ➔ 12 feature types from three categories (DNS, String, Role)
  - ◆ Compute *individual* and *relative* features for each type (24 features)
  - ◆ Continuous features are normalized on [0-1] and discretized
- ➔ Continuous relative feature values are computed by comparing the individual value of the resource to its parent's individual value
  - ◆ less, equal, or more
  - ◆ none for the root resource

DNS Features

- Domain Level
  - www.google.com has level 2
  - Divide by maximum allowed levels (126)
- Alexa Ranking
  - Divide the ranking by 1M
- Top-level Domain (TLD)
- Host Type

NEU SecLab

DNS Features (TLD)

Individual

Value	Example
none	IPs, Extensions
gen	*.com, *.org
gen-subdomain	*.us.com
cc	*.us, *.de, *.cn
cc-subdomain	*.co.uk, *.com.cn
cc-int	*.xn--p1ai (ru)
other	*.biz, *.info

Relative

Value	Example
none	root resource
{got,lost}-tld	Ext. → *.de, *.us → IP
gen-to-{cc,other}	*.org → {*.de, *.info}
cc-to-{gen,other}	*.uk → {*.com, *.biz}
other-to-{gen,cc}	*.info → {*.net, *.uk}
same-{gen,cc,other}	*.com → *.com
diff-{gen,cc,other}	*.info → *.biz

NEU SecLab

DNS Features (Type)

Individual

Value	Example
ipv6	2607:f0d0::::4
ipv4-private	192.168.0.1
ipv4-public	4.2.2.4
extension	Ext. Scripts
dns-sld	google.com
dns-sld-sub	www.google.com
dns-non-sld	abc.dyndns.org
dns-non-sld-sub	a.b.dyndns.org

Relative

Value	Example
none	root resource
same-site	w.google.com → ad.google.com
same-sld	1.dyndns.org → 2.dyndns.org
same-company	ad.google.com → www.google.de
same-eff-tld	bbc.co.uk → london.co.uk
same-tld	bbc.co.uk → london.uk
different	google.com → facebook.net

String Features

- Non-alphabetic characters
- Unique characters
- Character frequency
- Length
- Entropy

NEU SecLab

Role Features

- Three binary features
  - Ad network
  - Content delivery network (CDN)
  - URL shortening service
- Manually compiled list

NEU SecLab

Implementation

- Modifications to Chromium browser
  - Blink
  - Extension Engine
- ~1,000 SLoC (C++) and several lines of JavaScript
- Tracking the start and termination of JavaScript execution
- Tracking content scripts injection and execution
- Tracks network requests
- Callbacks registered for events and timers

Data Collection

NEU SecLab

- ➔ Alexa Top 200K from June 2014 to May 2015
- ➔ Alexa Top 20 shopping sites with 292 ad-injecting extensions for one week in June 16th-22nd, 2015
- ➔ Anti-cloaking
- ➔ Anti-fingerprinting

Item	Website Crawl	Extension Crawl
Websites Crawled	234,529	20
Unavailable Websites	7,412	0
Unique Inclusion Trees	47,789,268	35,004
Unique Inclusion Sequences	27,261,945	61,489
Unique URLs	546,649,590	72,064
Unique Domains	1,368,021	1,144
Unique Sites	459,615	749
Unique SLDs	419,119	723
Unique Companies	384,820	719
Unique Effective TLDs	1,115	21
Unique TLDs	404	21
Unique IPs	9,755	3

Building Labeled Dataset

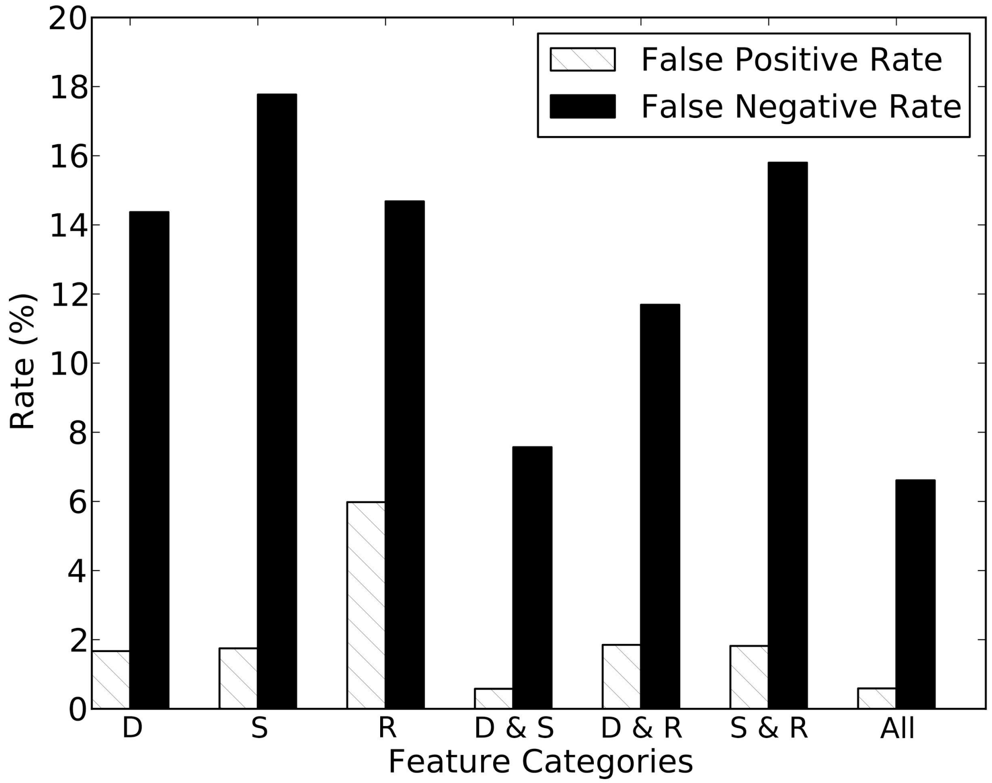
NEU SecLab

- ➔ Trained classifiers using VirusTotal as ground truth
  - ◆ host is reported malicious by at least 3 out of the 62 URL scanners

Dataset	No. of Inclusion Sequences		No. of Terminal Domains	
	Website Crawl	Extention Crawl	Website Crawl	Extension Crawl
Benign	3,706,451	7,372	35,044	250
Malicious	25,153	19	1,226	2

Detection Results

- 10-fold cross-validation
  - FP ⇒ 0.59%
  - FN ⇒ 6.61%
- Features effectiveness
  - D ⇒ DNS
  - S ⇒ String
  - R ⇒ Role

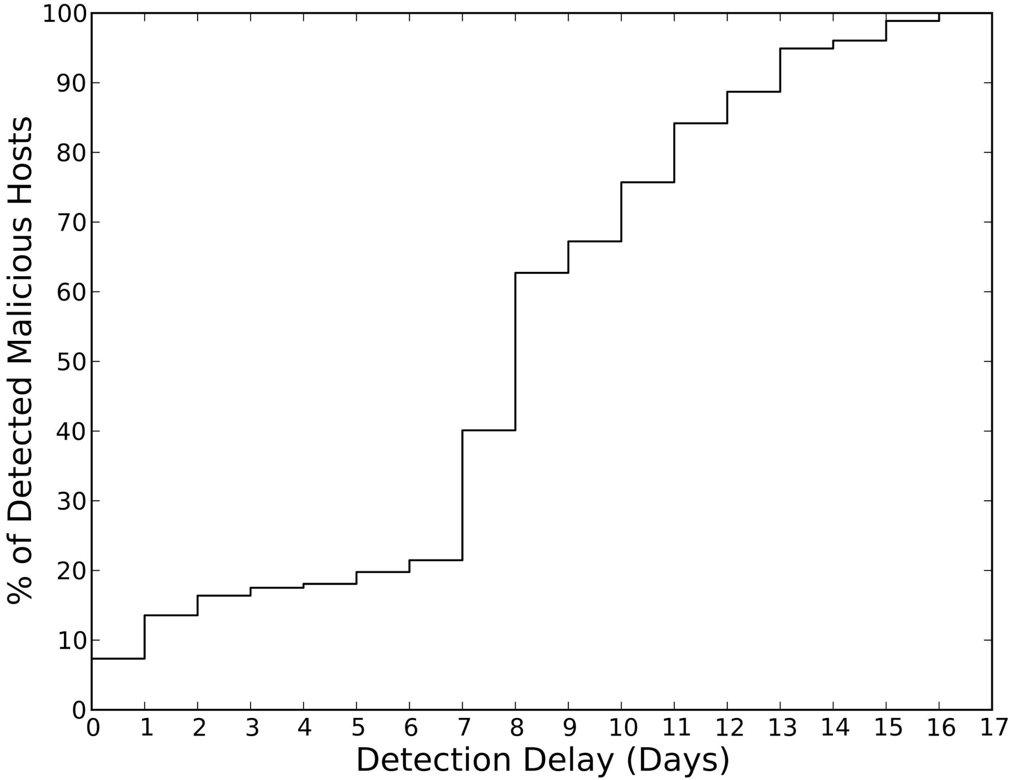


Comparison with URL Scanners

NEU SecLab

- ➔ Compared detection results on new data from June 1 to July 14, 2015
- ➔ Found 89 suspicious hosts that were likely dedicated redirectors
  - ◆ 44% were recently registered in 2015
  - ◆ 23% no longer resolve to an IP address
- ➔ Detected 177 new malicious hosts later reported in VT
  - ◆ 78% of the malicious hosts were not reported during the first week

Early Detection



Performance

- Automatically visited the Alexa Top 1K with original and modified Chromium browsers for 10 times
- Installed 5 popular Chrome extensions
  - Adblock Plus, Google Translate, Google Dictionary, Evernote WebClipper, and Tampermonkey
- Average 12.2% page latency overhead
- 3.2 seconds delay on browser startup time

NEU SecLab

Usability

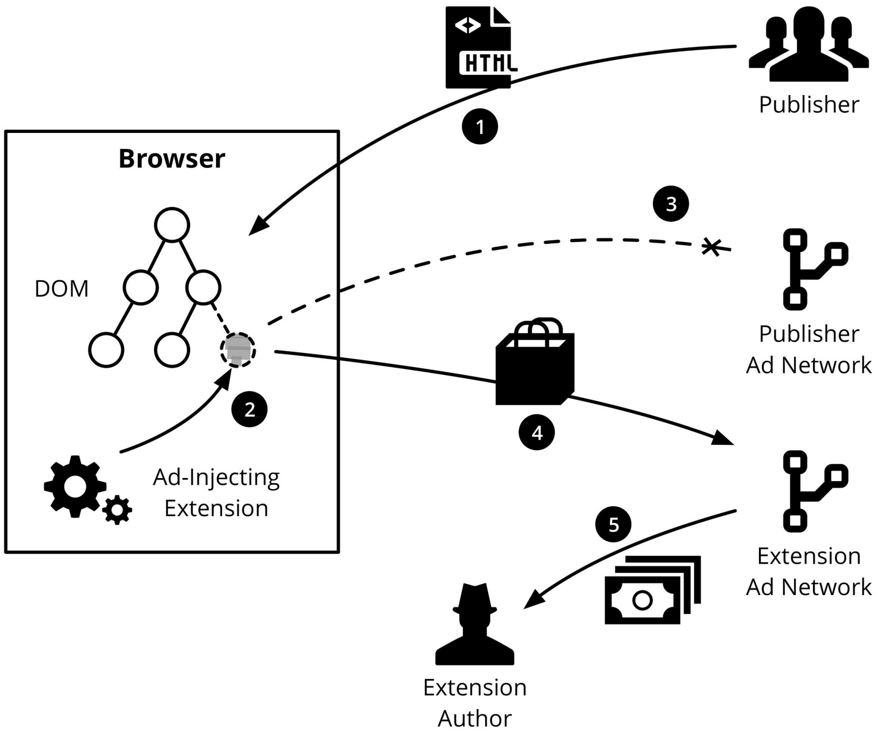
- 10 students that self-reported as expert Internet users
- Each participant explored 50 random websites from Alexa Top 500
    - Excluded websites requiring a login or involving sensitive subject matter
- Out of 5,129 web pages visited:
    - 31 malicious inclusions
    - 83 errors (mostly high latency resource loads)
- No broken extension was reported

NEU SecLab

OriginTracer

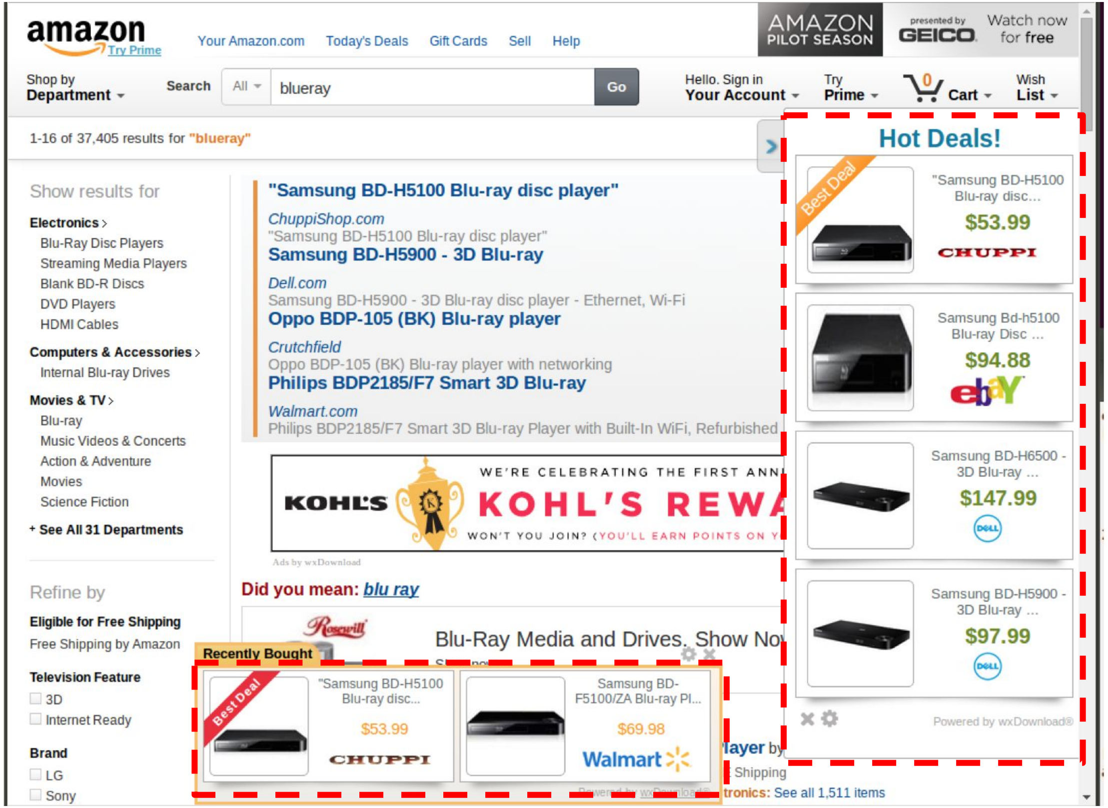
Identifying Ad Injection in Browser Extensions

Ad Injection



NEU SecLab

Ad Injection



Motivation

- ➔ Centralized dynamic analysis is non-trivial
  - ◆ Hiding behaviors during the analysis time, triggering ad injection
- ➔ Third-party content injection or modification is quite common
- ➔ Non-trivial to delineate between wanted and unwanted behavior

Users are best positioned to make this judgment

Extension	No. of Users	Injected Element
Adblock Plus	10,000,000+	`<iframe>`
Google Translate	6,000,000+	`<div>`
Tampermonkey	5,800,000+	`<img>`
Evernote Web Clipper	4,300,000+	`<iframe>`
Google Dictionary	3,000,000+	`<div>`

OriginTracer

OriginTracer adds fine-grained content provenance tracking to the web browser

- ➔ Provenance tracked at level of individual DOM elements
- ➔ Indicates origins contributing to content injection and modification
- ➔ Trustworthy communication of this information to the user

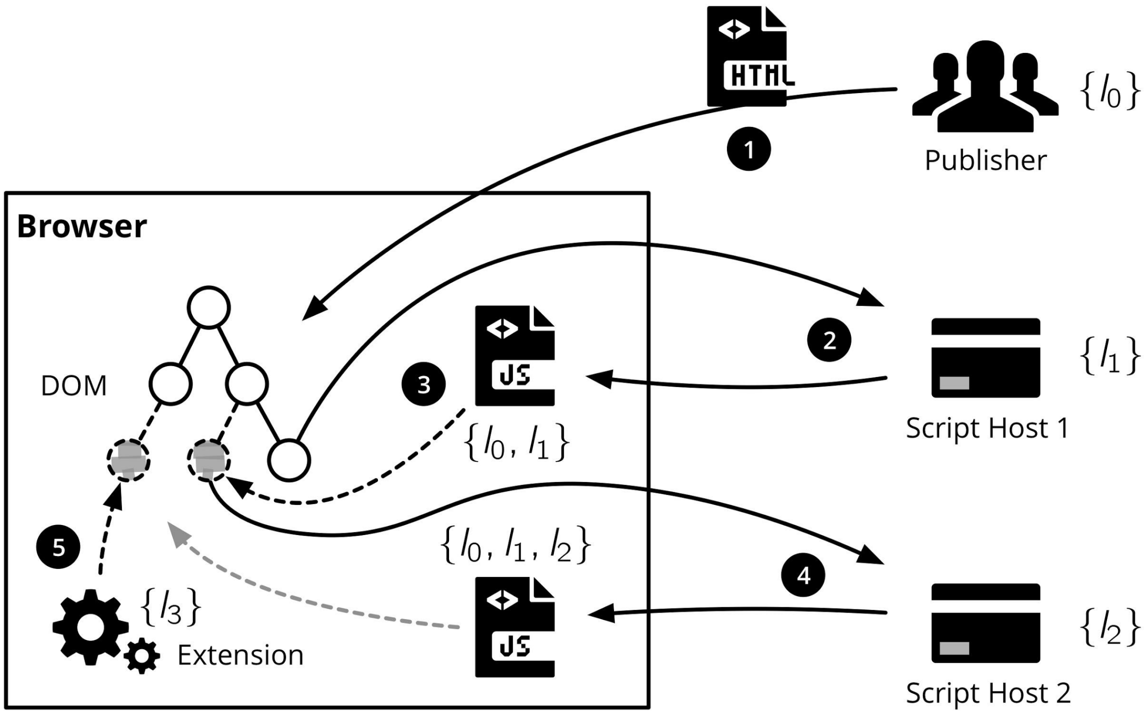
Provenance Labels

NEU SecLab

➔ Labels are generalizations of web origins

$$L = \langle S, I, P, X \rangle$$
$$S = \{\texttt{scheme}\} \cup \{\texttt{"extension"}\}$$
$$I = \{\texttt{host}\} \cup \{\texttt{extension-identifier}\}$$
$$P = \{\texttt{port}\} \cup \{\texttt{null}\}$$
$$X = \{0, 1, 2, \ldots\}$$

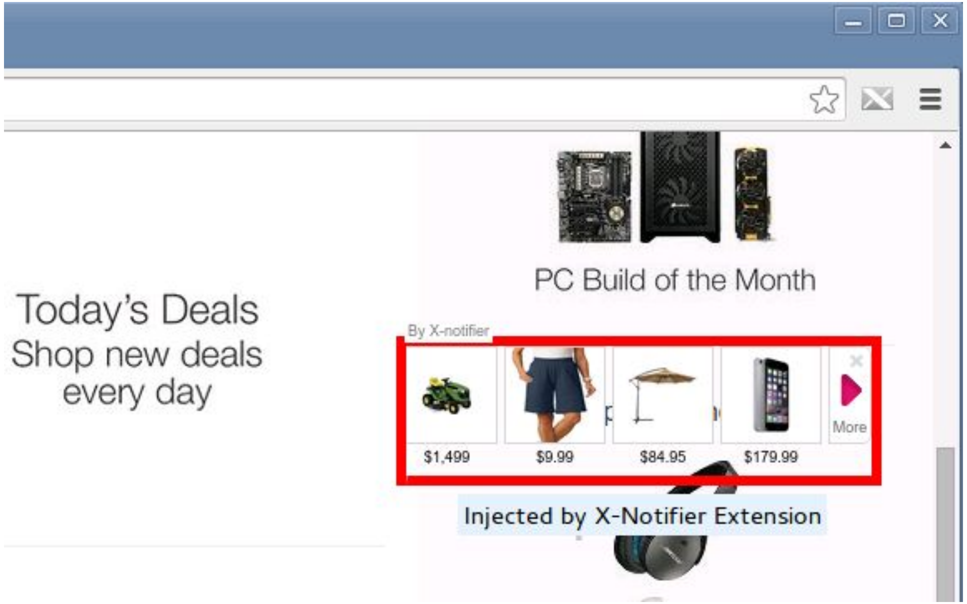
Label Propagation



Provenance Indicators

NEU SecLab

Provenance must be communicated to the user in a trustworthy and an easy-to-comprehend way



Implementation

- Modifications to Chromium browser
- ~900 SLoC (C++), several lines of JavaScript
- Mediates DOM APIs for node creation and modification
- Mediates node insertion through document writes
- Callbacks registered for events and timers

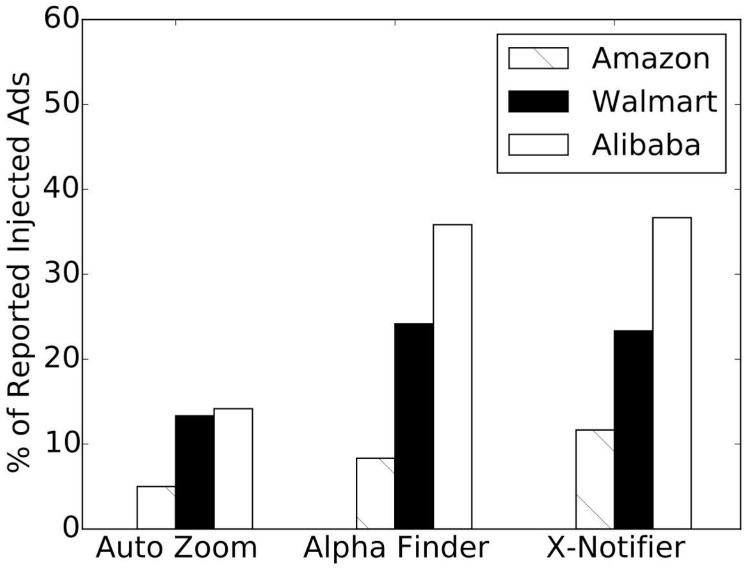
User Study Setup

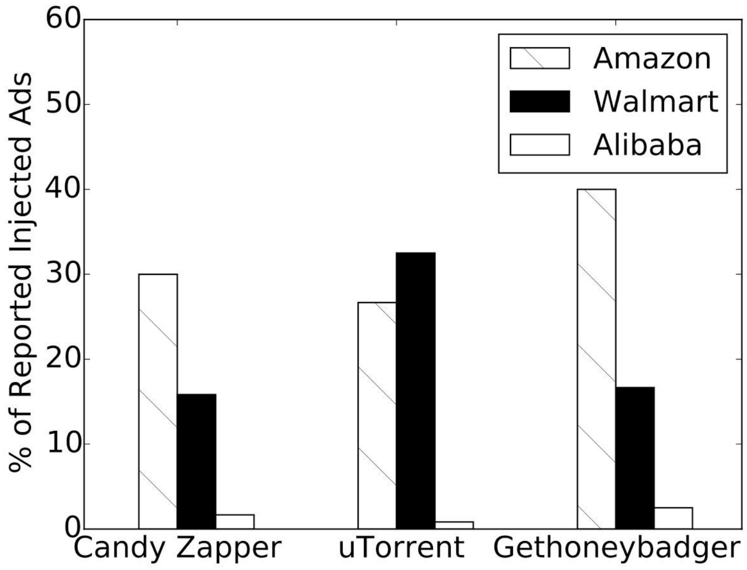
- Study population: 80 students of varying technical sophistication
- Participants exposed to six Chromium instances (unmodified and modified), each with an ad-injecting extension installed
  - Auto Zoom, Alpha Finder, X-Notifier, Candy Zapper, uTorrent, Gethoneybadger
- Participants were asked to visit three retail websites
  - Amazon, Walmart, Alibaba

Reported Injected Ads

NEU SecLab

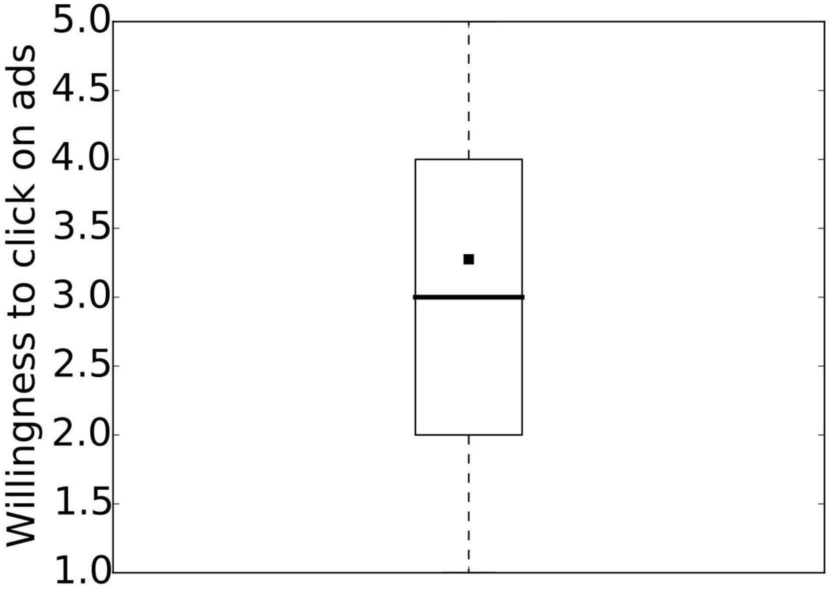
Are users able to correctly recognize injected advertisements?





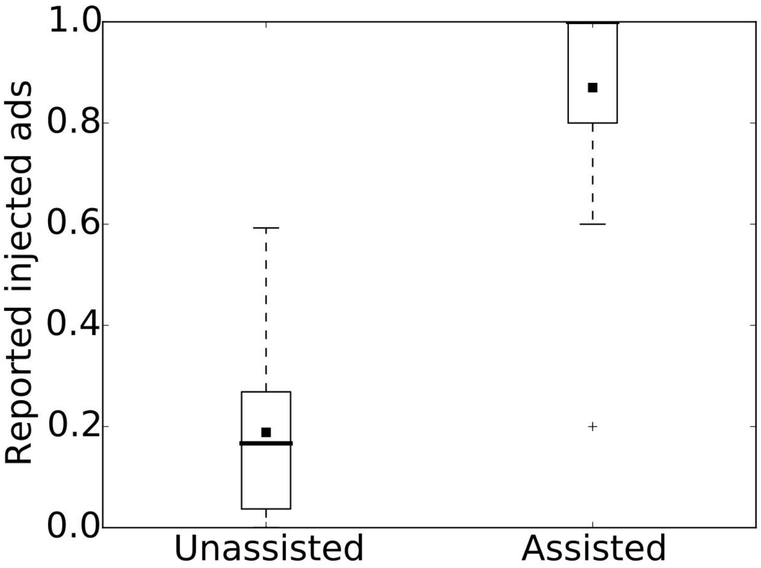
Susceptibility to Ad Injection

Are users generally willing to click on the advertisements presented to them?



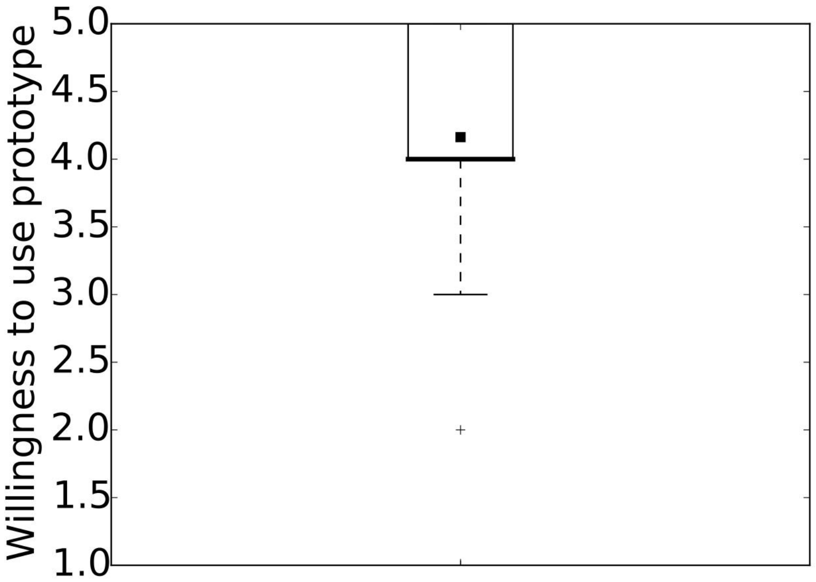
Ability to Identify Injected Ads

Do content provenance indicators assist users in recognizing injected advertisements?



Usability of Content Provenance

Would users be willing to adopt a provenance tracking system to identify injected advertisements?



Performance

NEU SecLab

- Configured an unmodified Chromium and OriginTracer instance to visit the Alexa Top 1K
  - Broad spectrum of static and dynamic content on most-used websites
  - Browsers configured with five benign extensions
- Average 10.5% browsing latency overhead
- No impact on browser start-up time

Usability

NEU SecLab

- Separate user study on 13 students of varying technical background
- Asked participants to browse 50 websites out of Alexa Top 500
- Asked users to report errors
  - Type I: browser crash, page doesn’t load, etc.
  - Type II: abnormal load time, page appearance not as expected
- Out of almost 2K URLs, two Type I and 27 Type II errors were reported
- No broken extensions was reported

Analysis of Style Injection by Relative Path Overwrite (RPO)

Relative Path Overwrite (RPO)

NEU SecLab

- ➔ Browser's interpretation of URL may be different than the web server
  - ◆ Browsers basically treat URLs as file system-like paths
  - ◆ *However, URL may not correspond to an actual server-side file system structure, or web server may internally rewrite parts of the URL*
- ➔ RPO exploits the semantic disconnect between browsers and web servers in interpreting relative paths ⇒ ***path confusion***
  - ◆ Injects style (CSS) instead of script (JS)
  - ◆ Turns a simple *text injection* vulnerability into a *style sink*
  - ◆ **"Self-reference": Attacked document uses "itself" as stylesheet**
- ➔ Threat model of RPO resembles that of XSS
  - ◆ e.g., steal sensitive information

NEU SecLab

Path Confusion

Web Page: **http://example.com/dir/page.php**
Relative Style: **files/style.css**
Absolute Style: **http://example.com/dir/files/style.css**

Path Confusion

NEU SecLab

Web Page: **http://example.com/dir/page.php**
Relative Style: **files/style.css**
Absolute Style: **http://example.com/dir/files/style.css**

Web Page: **http://example.com/dir/page.php/**
Relative Style: **files/style.css**
Absolute Style: **http://example.com/dir/page.php/files/style.css**

NEU SecLab

Style Injection

Browser Server

http://example.com/***{background-image:url(...)}**/

NEU SecLab

Style Injection

Browser → Server

```
http://example.com/*{background-image:url(...)}/
```

Server → Browser

```
<html>
 <head>
  <link rel="stylesheet" href="style.css">
 </head>
 <body>
  Not found:
  http://example.com/*{background-image:url(...)}/
 </body>
</html>
```

Self-reference

Browser   Server

`http://example.com/*{background-image:url(...)}/`

Self-reference

NEU SecLab

Browser   Server

`http://example.com/*{background-image:url(...)}/` →

← `...<link rel="stylesheet" href="style.css">...`

Self-reference

NEU SecLab

Browser Server

http://example.com/*{background-image:url(...)}/

...<link rel="stylesheet" href="**style.css**">...

http://example.com/*{background-image:url(...)}/**style.css**

NEU SecLab

Self-reference

Browser   Server

http://example.com/*{background-image:url(...)}/

...<link rel="stylesheet" href="**style.css**">...

http://example.com/*{background-image:url(...)}/**style.css**

```
<html>
 ...
 Not found:
 http://example.com/*{background-image:url(...)}/style.css
 ...
</html>
```

Scriptless (Style-based) Attacks

- Script injection is *NOT* always possible
  - Input sanitization
  - Browser-based XSS filters
  - Content Security Policy (CSP)
- Successful attacks are possible by injecting CSS
  - Exfiltrating credit card number and CSRF tokens (Heiderich et al., CCS 2012)
    - CSS attribute accessor, content property, animation features, media queries
- Attacks typically consist of *payload* & *injection* technique
- Our work is not concerned about the payload
  - Focus on how to inject (“transport”) the payload

Preconditions for Successful Attack

1. Relative stylesheet path (no base tag) ⇒ **Candidate Identification**
2. Crafted URL causes style reflection in server response ⇒ **Vulnerability Detection**
3. Browser loads and interprets injected style directives ⇒ **Exploitability Detection**

NEU SecLab

Candidate Identification

- ➔ Common Crawl: extract pages with relative stylesheet path
  - August 2016: >1.6B documents
  - 203 M pages on nearly 6 M sites
- ➔ Filter 1: Alexa Top 1 million ranking
  - 141 M pages on 223 K sites
- ➔ Filter 2: Group URLs using the same template
  - Test only one random URL from each group
  - 137 M pages on 222 K sites

Group By	**URL**
Query Parameter	http://example.com/?lang=**en** http://example.com/?lang=**fr**
Path Parameter	http://example.com/**028** http://example.com/**142**

Vulnerability Detection

NEU SecLab

- ➔ Developed a lightweight crawler based on Python Requests API
  - ◆ Randomly selects one representative URL from each group
  - ◆ Mutates the URL according to a number of path confusion techniques
    - `PAYLOAD` ⇒ `%0A{}body{background:NONCE}`
  - ◆ Requests the mutated URL
  - ◆ Ignores the response if it contains <base> tag
  - ◆ Extracts all relative stylesheet paths and expands them using the mutated URL
  - ◆ requests each expanded stylesheet URL to find injected payload in the response
  - ◆ Page would be vulnerable if at least one stylesheet's response reflects the requested URL, referrer URL, parameters, or cookie
- ➔ Path confusion techniques
  - ◆ Path Parameter, Encoded Path, Encoded Query, Cookie
  - ◆ We assume the page references relative stylesheet path **../style.css**

Path Confusion - Path Parameter

NEU SecLab

- ➔ Some web frameworks (e.g., PHP, ASP, JSP) accept the input parameters as a directory-like string following the name of the script in the URL
- ➔ Simply appends the payload as a subdirectory to the end of the URL
- ➔ CSS injection occurs if the page reflects page URL or referrer in the response

```
http://domain/dir/page.asp
http://domain/dir/page.asp/PAYLOAD//
http://domain/dir/page.asp/PAYLOAD/style.css
```

Path Confusion - Path Parameter

NEU SecLab

- Different web frameworks handle path parameters differently, which is why we distinguish a few additional cases
  - parameters separated by slashes in PHP/ASP and semicolons in JSP

```
http://domain/page.php/param
http://domain/page.php/PAYLOADparam//
http://domain/page.php/PAYLOADparam/style.css
```

```
http://domain/dir/page.jsp;param
http://domain/dir/page.jsp;PAYLOADparam//
http://domain/dir/page.jsp;PAYLOADparam/style.css
```

Path Confusion - Encoded Path

NEU SecLab

- ➔ This targets web servers such as IIS that decode encoded slashes in the URL for directory traversal, whereas web browsers ***DO NOT***
- ➔ Use **%2F**, an encoded version of **'/'**, to inject our payload into the URL
- ➔ The canonicalized URL is equal to the original page URL
- ➔ CSS injection occurs if the page reflects page URL or referrer in the response

```
http://domain/dir/page.aspx
http://domain/PAYLOAD/..%2Fdir/PAYLOAD/..%2Fpage.aspx//
http://domain/PAYLOAD/..%2Fdir/PAYLOAD/..%2Fpage.aspx/style.css
```

Path Confusion - Encoded Query

NEU SecLab

- ➔ We replace the URL query delimiter **'?'** with its encoded version **%3F** so that web browsers **DO NOT** interpret it as such
- ➔ We inject the payload into every value of the query string
- ➔ CSS injection happens if the page reflects either the URL, referrer, or any of the query values in the HTML response

```
http://domain/dir/page.html?key=value
http://domain/dir/page.html%3Fkey=PAYLOADvalue//
http://domain/dir/page.html%3Fkey=PAYLOADvalue/style.css
```

Path Confusion - Cookie

NEU SecLab

- Stylesheets referenced by a relative path are located in the same origin
  - Cookies are sent when requesting the stylesheet
- CSS injection may be possible if:
  - Attacker can create new cookies or tamper with existing ones, and
  - The page reflects cookie values in the response
- Payload is injected into each cookie value

```
http://domain/dir/page.php?key=value
http://domain/dir/page.php//?key=value
http://domain/dir/page.php/style.css

Original Cookie: name=val
 Crafted Cookie: name=PAYLOADval
```

Exploitability Detection

NEU SecLab

- ➔ Verify whether the reflected CSS in the response is evaluated by the browser
  - ◆ Built a crawler based on Google Chrome (and an extension for tainting cookie)
- ➔ Visit mutated vulnerable pages to check if injected style directives interpreted
  - ◆ `PAYLOAD ⇒ %0A}}}]]]{}body{background-image:url(/NONCE.gif)}`
  - ◆ Style is interpreted if injected image URL seen in network traffic
- ➔ Reflected CSS is not always interpreted by the browser
  - ◆ Use of modern document types ⇒ browser doesn’t render page in quirks mode
- ➔ Overriding document types in Internet Explorer (IE)
  - ◆ Load the page inside an iframe in Internet Explorer
  - ◆ Used Fiddler for tainting cookies and recording HTTP requests/responses
  - ◆ Turn on Compatibility View by setting “X-UA-Compatible” to “IE=EmulateIE7” via <meta> tag in the parent page

Limitations

NEU SecLab

- ➔ We only looked for *reflected*, not stored, injection of style directives
- ➔ Manual analysis of a site might reveal more opportunities for style injection that our crawler fails to detect automatically
- ➔ We did not analyze the vulnerability of pages not in Common Crawl
  - ◆ We do not cover all sites, and we do not fully cover all pages within a site
- ➔ Results presented in this paper should be seen as a lower bound
- ➔ Our methodology can be applied to individual sites to analyze more pages

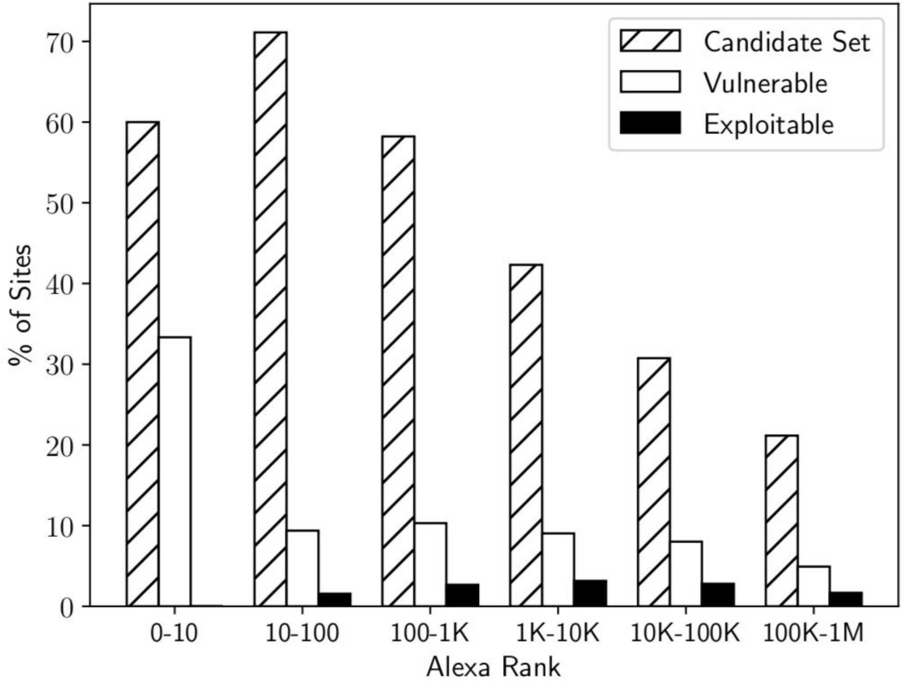
Dataset

	Relative CSS	Alexa Top 1M	Candidate Set
All Pages	203,609,675	141,384,967	136,793,450
Tested Pages	53,725,270	31,448,446	30,991,702
Sites	5,960,505	223,212	222,443
Doc. Types	9,833	2,965	2,898

Alexa Ranking

NEU SecLab

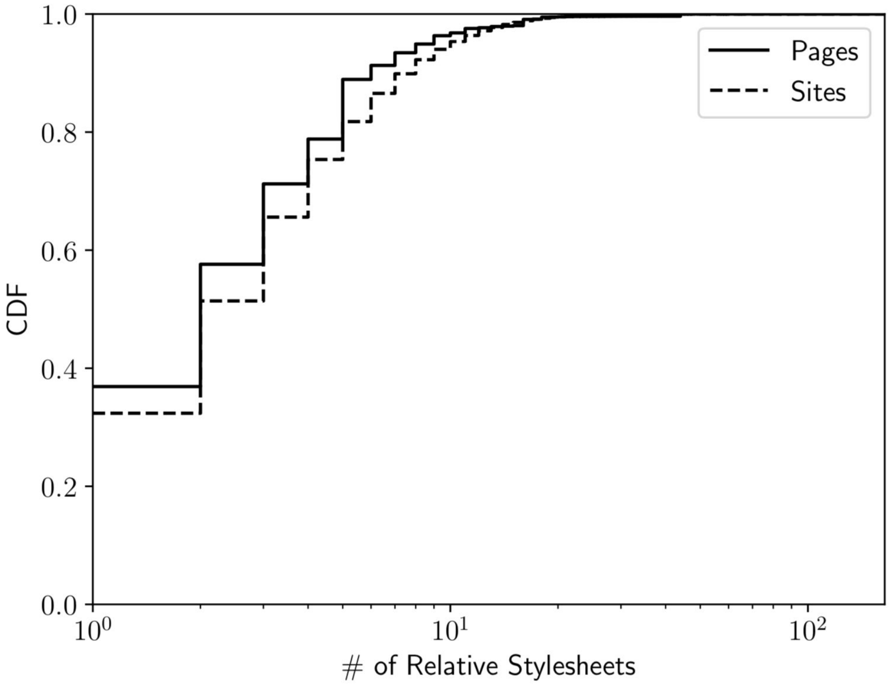
- ➔ Six out of the ten largest sites are represented in our candidate set
- ➔ Candidate set contains a higher fraction of the largest sites and a lower fraction of the smaller sites
- ➔ Our results better represent the most popular sites, which receive most visitors, and most potential victims of RPO attacks



Relative Stylesheet Paths

NEU SecLab

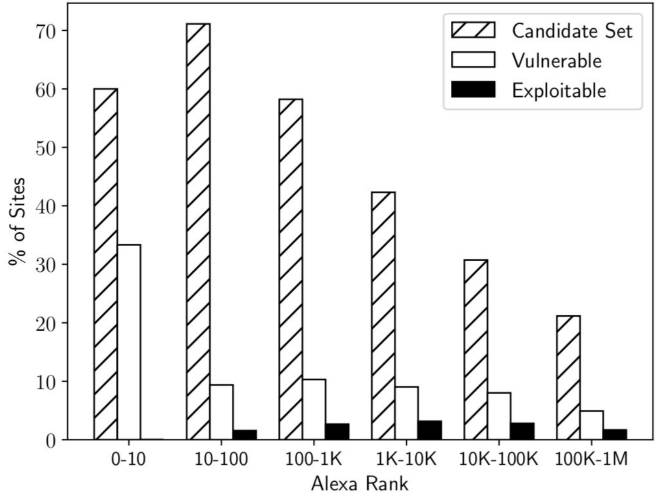
- CDF of total and maximum number of relative stylesheets per web page and site, respectively
- **63.1%** of the pages contain multiple relative paths
  - Increases the chances of finding a successful RPO and style injection point



Vulnerability Analysis

NEU SecLab

- A page is vulnerable if its response:
  - Reflects the injected CSS
  - Does not contain a base tag
- **1.2%** of pages are vulnerable to at least one of the injection techniques
- **5.4%** of sites contain at least one vulnerable page
- Path parameter is the most effective technique against pages
- **One third** of the sites in Alexa Top 10, **8-10%** in the Top 100K, and **4.9%** in 100K-1M are vulnerable



Technique	Pages	Sites
Path Parameter	309,079 (1.0%)	9,136 (4.1%)
Encoded Path	53,502 (0.2%)	1,802 (0.8%)
Encoded Query	89,757 (0.3%)	1,303 (0.6%)
Cookie	15,656 (<0.1%)	1,030 (0.5%)
Total	377,043 (1.2%)	11,986 (5.4%)

Base Tag

NEU SecLab

- Correctly configured base tag can prevent path confusion
- Base tag was removed after path confusion in **603 pages** on **60 sites**
- Internet Explorer fetches two URLs for stylesheet
  - One expanded according to the base URL specified in the tag
  - One expanded using the page URL as the base
- Internet Explorer always applied the *“confused”* stylesheet, even when the one based on the base tag URL loaded faster

Quirks Mode Doc. Types

NEU SecLab

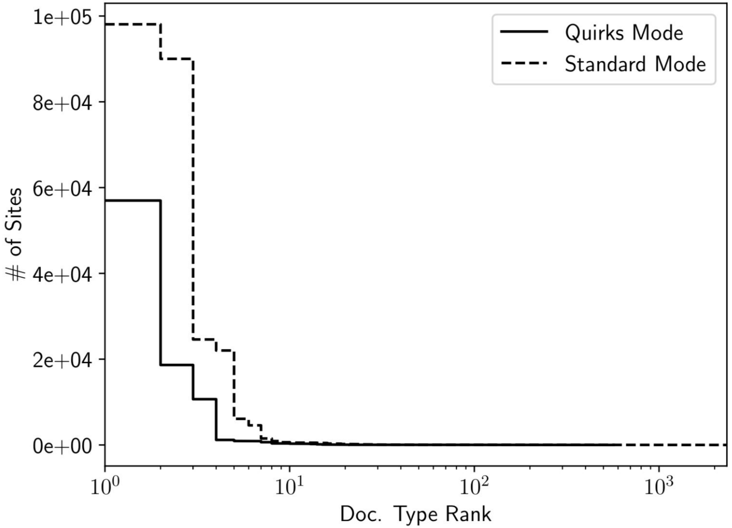
- Browsers parse resources with a non-CSS content type when the page specifies a non-standard document type (or none at all)
- Total of **4,318** distinct doc. Types
- Roughly a third result in quirks mode
  - **1,271 (29.4%)** force all the browsers into quirks mode
  - **1,378 (31.9%)** cause at least one browser to use quirks mode
- Internet Explorer's framing trick forced **4,248 (98.4%)** into quirks mode

Browser	Version	OS	Doc. Types
Chrome	55	Ubuntu 16.04	1,378 (31.9 %)
Opera	42	Ubuntu 16.04	1,378 (31.9 %)
Safari	10	macOS Sierra	1,378 (31.9 %)
Firefox	50	Ubuntu 16.04	1,326 (30.7 %)
Edge	38	Windows 10	1,319 (30.5 %)
IE	11	Windows 7	1,319 (30.5 %)

Doc. Type (shortened)	Pages	Sites
(none)	1,818,595 (5.9 %)	56,985 (25.6 %)
"-//W3C//DTD HTML 4.01 Transitional//EN"	721,884 (2.3 %)	18,648 (8.4 %)
"-//W3C//DTD HTML 4.0 Transitional//EN"	385,656 (1.2 %)	11,566 (5.2 %)
"-//W3C//DTD HTML 3.2 Final//EN"	22,019 (<0.1 %)	1,175 (0.5 %)
"-//W3C//DTD HTML 3.2//EN"	10,839 (<0.1 %)	927 (0.4 %)
All	3,046,449 (9.6 %)	71,597 (32.2 %)

Standardized Doc. Types

- ~1K doc. types result in quirks mode
- ~3K doc. types cause standards mode
- But, number of standardized doc. types is several orders of magnitude smaller
  - Only about **10** standards and quirks mode doc. types are widely used in sites
  - Majority are not standardized
  - Differ from the standardized ones only by small variations such as forgotten spaces or misspellings
- 9.6% of pages use quirks modes
- 32.2% of sites contain at least one page rendered in quirks mode

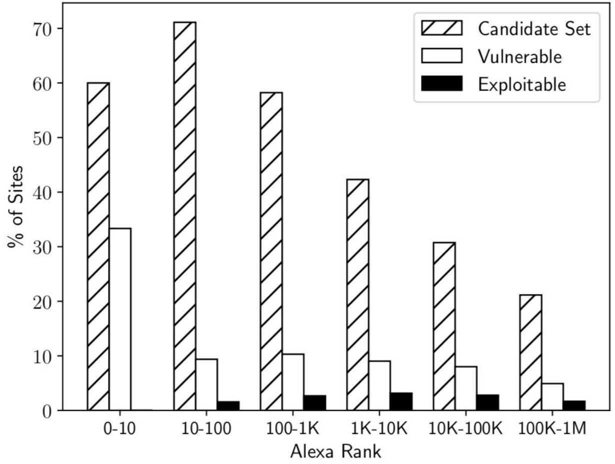


Doc. Type	At Least One Page	All Pages
None	56,985 (25.6%)	19,968 (9.0%)
Quirks	27,794 (12.5%)	7,720 (3.5%)
None or Quirks	71,597 (32.2%)	30,040 (13.5%)
Standards	192,403 (86.5%)	150,846 (67.8%)

Exploitability Analysis

NEU SecLab

- ➔ Vulnerable pages that were exploitable
  - ◆ 2.9% in Chrome
  - ◆ 14.5% in Internet Explorer
    - **5x** more than in Chrome
- ➔ 6 highest-ranked sites were not exploitable
  - ◆ Between 1.6% and 3.2% of sites in the remaining buckets were exploitable
- ➔ IE is more effective *except in cookie*
  - ◆ IE crawl was conducted one month later
  - ◆ **Anti-framing** techniques
  - ◆ **Anti-MIME-sniffing** header



Technique	Chrome		Internet Explorer	
	Pages	Sites	Pages	Sites
Path Parameter	6,048 (<0.1%)	1,025 (0.5%)	52,344 (0.2%)	3,433 (1.5%)
Encoded Path	3 (<0.1%)	2 (<0.1%)	24 (<0.1%)	5 (<0.1%)
Encoded Query	23 (<0.1%)	20 (<0.1%)	137 (<0.1%)	43 (<0.1%)
Cookie	4,722 (<0.1%)	81 (<0.1%)	2,447 (<0.1%)	238 (0.1%)
Total	10,781 (<0.1%)	1,106 (0.5%)	54,853 (0.2%)	3,645 (1.6%)

NEU SecLab

Anti-Framing

1. **X-Frame-Options** response header (DENY, SAMEORIGIN, or ALLOW-FROM)
   - **4,999 vulnerable pages** on **391 sites** used it *correctly* and prevented the attack
   - However, **1,900 pages** on **34 sites** provided *incorrect values* for this header
     - Out of which, **552 pages** on **2 sites** were exploited in Internet Explorer
2. **frame-ancestors** directive in Content Security Policy (not supported in IE)
   - A whitelist of origins allowed to load the current page in a frame
3. Use JavaScript code to prevent framing of a page
   - i.e., redirecting the top frame if the page is not the top window itself
   - However, attackers can use the HTML5 sandbox attribute in the iframe tag and omit the allow-top-navigation directive to render JavaScript frame-busting code ineffective

We did not implement any of these techniques to allow framing, which means that more vulnerable pages could likely be exploited in practice

NEU SecLab

MIME Sniffing

- Many sites contain misconfigured content types
  - Browsers attempt to infer the type based on the request context or file extension
    - *MIME sniffing*, especially in quirks mode
- Setting **X-Content-Type-Options: nosniff** in response header block the request if the requested type is:
  - "style" and the MIME type is not "text/css", or
  - "script" and the MIME type is not, i.e., “application/javascript”
- Only Firefox, Internet Explorer, and Edge respected this header *at the time*
  - Chrome started supporting the header since January 2018
  - IE blocked our injected CSS payload when **nosniff** was set even with framing trick
- **96,618 vulnerable pages** across **232 sites** had a nosniff response header
  - **23 pages** across **10 sites** were exploitable in Chrome but not in Internet Explorer

NEU SecLab

Content Management Systems

- ➔ Many exploitable pages appeared to belong to well-known CMSes
  - ◆ CMSes are installed on thousands of sites, fixing RPO vulnerability is impactful
- ➔ Detected **23 CMSes** (Wappalyzer + manually)
  - ◆ **41,288 pages** across **1,589 sites**
- ➔ Installed the latest versions (or used the online demos)
- ➔ Detected **4 exploitable** CMSes
  - ◆ **1** declared no document type
  - ◆ **1** used a quirks mode document type
  - ◆ **2** were exploited in IE using framing trick
  - ◆ **40,255 pages** across **1,197 sites** (nearly 32k sites world-wide)
- ➔ Weaknesses were reported to the vendors

NEU SecLab

Mitigation Techniques

- Avoid *path confusion*
  - Use only **absolute (or root-relative) paths** or **<base> tag**
- Avoid *style injection*
  - Input sanitization (non-trivial)
    - More targeted RPO attack variants can reference existing files
- Prevent stylesheets with syntax errors or no "text/css" content type
  - Specify a modern document type: **<!doctype html>**
  - Disable content type sniffing: **X-Content-Type-Options**
- Prevent Internet Explorer trick
  - Disallow framing: **X-Frame-Options**
  - Turn off compatibility view: **X-UA-Compatible (IE=Edge)**

NEU SecLab

Conclusion

- Excision
  - Allows for preemptive blocking with moderate performance overhead
  - Detected malicious hosts before appearing in the blacklists
- OriginTracer
  - Allows users to make fine-grained trust decisions
  - Evaluation shows it can be performed in an efficient and effective way
- RPO
  - Style-based attacks require different countermeasures than XSS
  - Easy-to-use and effective countermeasures exist to mitigate the attack

Thesis Publications

- **Third-party Content Inclusion ⇒ Excision**
  - **Sajjad Arshad**, Amin Kharraz, William Robertson, "Include Me Out: In-Browser Detection of Malicious Third-Party Content Inclusions", *Financial Cryptography and Data Security (FC), 2016*
- **Ad Injection ⇒ OriginTracer**
  - **Sajjad Arshad**, Amin Kharraz, William Robertson, "Identifying Extension-based Ad Injection via Fine-grained Web Content Provenance", *Research in Attacks, Intrusions and Defenses (RAID), 2016*
- **Relative Path Overwrite (RPO)**
  - **S. Arshad**, Seyed Ali Mirheidari, Tobias Lauinger, Bruno Crispo, Engin Kirda, William Robertson, "Large-Scale Analysis of Style Injection by Relative Path Overwrite", *The Web Conference (WWW), 2018*

NEU SecLab

Deep Crawling

- The *Inclusion Tree* crawler has been evolving
  - Written in NodeJS
  - Uses Chrome Remote Debugging protocol
  - Publicly available (https://github.com/sajjadium/DeepCrawling)
- **Web Security**
  - Tobias Lauinger, Abdelberi Chaabane, **Sajjad Arshad**, William Robertson, Christo Wilson, Engin Kirda, "Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web", *Network and Distributed System Security Symposium (NDSS), 2017*

Deep Crawling

NEU SecLab

➔ **Tracking and Privacy**

- Muhammad Ahmad Bashir, **Sajjad Arshad**, William Robertson, Christo Wilson, "Tracing Information Flows Between Ad Exchanges Using Retargeted Ads", *USENIX Security Symposium, 2016*
- Muhammad Ahmad Bashir, **Sajjad Arshad**, Christo Wilson, "Recommended For You: A First Look at Content Recommendation Networks", *ACM Internet Measurement Conference (IMC), 2016*
- Muhammad Ahmad Bashir, **Sajjad Arshad**, Engin Kirda, William Robertson, Christo Wilson, "How Tracking Companies Circumvented Ad Blockers Using WebSockets", *ACM Internet Measurement Conference (IMC), 2018*

Other Works

NEU SecLab

- **Malware Detection**
  - Amin Kharraz, **Sajjad Arshad**, Collin Muliner, William Robertson, Engin Kirda, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware", *USENIX Security Symposium, 2016*
- **Binary Analysis**
  - Reza Mirzazade farkhani, Saman Jafari, **Sajjad Arshad**, William Robertson, Engin Kirda, Hamed Okhravi, "On the Effectiveness of Type-based Control Flow Integrity", *Annual Computer Security Applications Conference (ACSAC), 2018*

Thanks! Questions?

Sajjad Arshad

*https://**sajjadium**.github.io/*