Large-Scale Analysis of Style Injection by Relative Path Overwrite

Sajjad Arshad, Seyed Ali Mirheidari, Tobias Lauinger,
Bruno Crispo, Engin Kirda, William Robertson

WWW - 26 April 2018

Relative Path Overwrite

- Initially described in blog posts by Gareth Heyes and others
- Largely unknown type of web attack
  - Does not require markup injection or particular sink type
  - Inject style (CSS) instead of script (JS)
  - “Self-reference”: Attacked document uses “itself” as stylesheet
    (other attack variants exist, not studied in this work)
- Measure vulnerability surface of websites
  - How many websites can be exploited?
  - Which factors allow/prevent the attack?

Style Injection

Browser Server

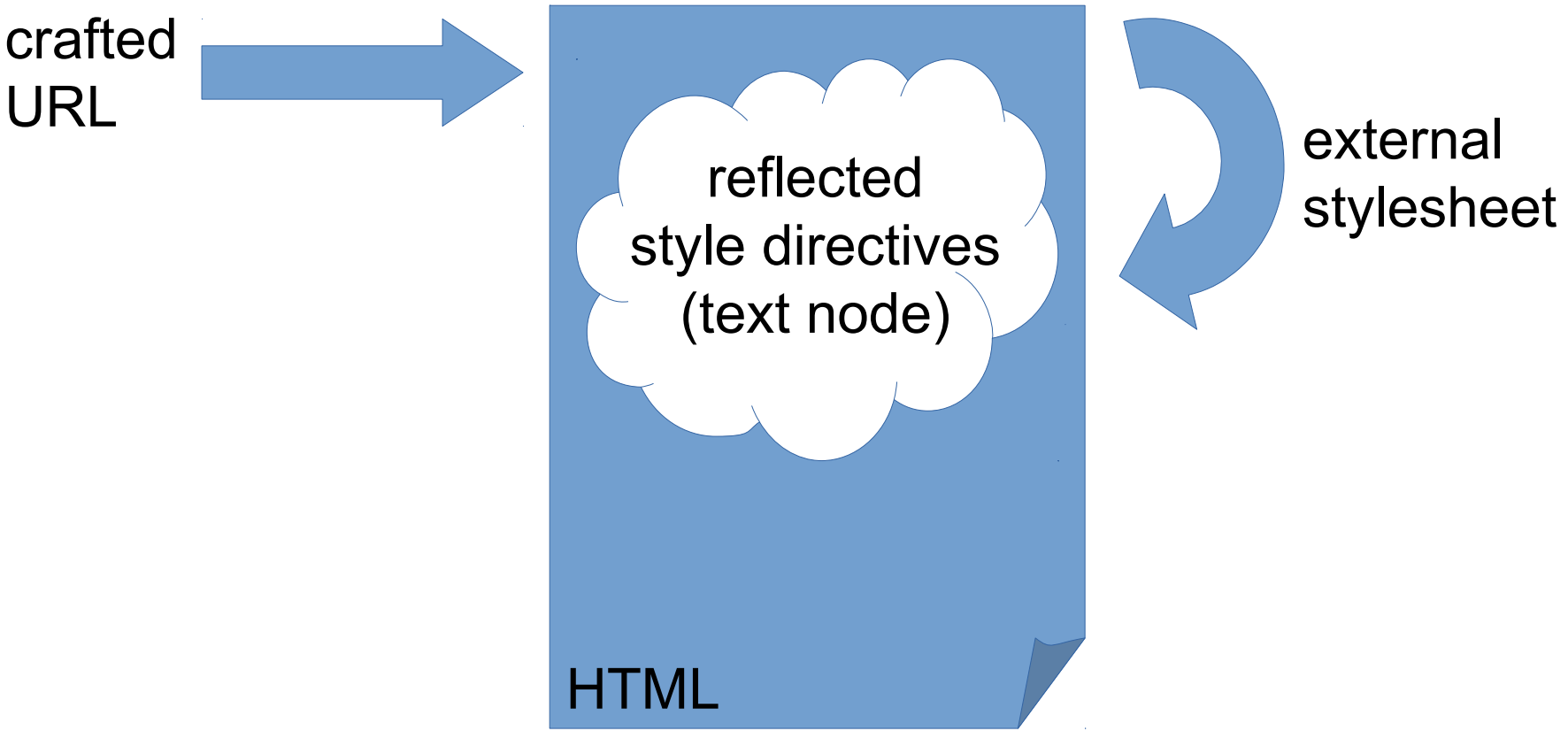
http://example.com/*{background-image:url(...)}/ ►

◄

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

Path Confusion

Self-Reference



Outline

- Relative Path Overwrite Attack
  - Reflection sink that allows injection of style directives
  - Path confusion causes stylesheet self-reference
- Measurement methodology
- Exploitability in practice
- Countermeasures

Style-Based Attacks

- Script injection not always possible (e.g., input sanitisation)
- Successful (exfiltration) attacks possible without script:
  - History stealing
  - Exfiltrate credit card number (Heiderich et al., CCS 2012)
- Attacks typically consist of payload & injection technique
  - This research is not concerned about the payload
  - Focus on how to inject (“transport”) the payload

Self-Reference

- Style injection requires either:
  - Markup injection (create a new <style> context), or
  - Correct sink type (injection inside existing <style> context)
- Both can be hard to find
- New technique:
  - Cause HTML document to reference “itself” as the stylesheet
  - Sink in document becomes style sink

Measurement Methodology

1. Common Crawl: extract pages with relative stylesheet path
   - August 2016: >1.6B documents
   - Filter: Alexa Top 1M, must contain relative stylesheet reference
2. Python script to test path confusion, if style can be injected
   - Try several URL mutation techniques (see paper)
   - Only GET, no clicking/form submission, benign payload
3. Browser to check if injected style directives interpreted
   - Open page in Chrome (+Internet Explorer with framing, see paper)
   - Style interpreted if injected image URL seen in network traffic

Preconditions for Successful Attack

- Relative stylesheet path: 13% of docs in Common Crawl
- Mutated URL causes style reflection in server response
  (and no base tag)
  - 1.2% of documents in Candidate Set
  - 5.4% of domains have at least one such document
- Browser loads and interprets injected style directives
  - Chrome: 11k documents (<0.1%) on 1k domains (0.5%)
  - IE: 55k documents (0.2%) on 4k domains (1.6%) → see paper

Injected Style Not Interpreted

- Only 2.9% of sites with injection successfully attacked
- Attack works only if document rendered in “quirks mode”
  - HTML “stylesheet” has wrong MIME type & lots of syntax errors
  - 32% of unique document types in Chrome result in quirks mode
  - <10% of documents (32% of domains) use quirks mode
- IE: can override document type with framing technique
  - Fails if anti-framing techniques used by site
  - Fails if content type sniffing disabled
  - see paper for details

Countermeasures (1)

Relative stylesheet path and path confusion → self-reference

- Browsers do not know about server-side content routing logic and assume directory-like URL structure
  - Use only absolute paths, or
  - Use `<base>` tag to tell browser how to resolve relative paths
- If absolute paths or base tag value dynamically computed on server side, must take into account actual routing logic

Countermeasures (2)

Text injection "vulnerability" and browser tolerating lots of CSS syntax errors → style injection

- Always escape any characters not in `[a-z0-9]` (is this realistic?)
- Prevent browsers from interpreting CSS files with many syntax errors and content type other than `text/css`
  - Specify a modern document type: `<!doctype html>`
  - In Internet Explorer, document type can be overwritten by framing, so set these headers:

```
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
```

Conclusion

- Over 5% of studied domains allow style injection and self-reference
- Much fewer domains can be exploited, but still a consequential number in absolute terms
- Different techniques (e.g., style-based attacks) require different countermeasures than script-based XSS
- In contrast to script-based XSS, easy-to-use and effective countermeasures exist to mitigate the attack