How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security

Ben Stock, Martin Johns, Marius Steffens, and Michael Backes
USENIX Security 2017
Motivation...

• Web's client side becomes more powerful every day
  • grew from static HTML rendering to fully-fledged applications
  • many "enabling" APIs such as postMessages

• Development also carries security issues
  • specific to the Web, e.g., XSS
  • general issues: e.g., trusting data from untrusted sources

• Web grew without a security blueprint into the "Tangled Web"
... and Research Questions

- **Goal: evaluate how web and security evolved**
  - What were most prevalent technologies over time?
  - Which security issues surfaced over time?
  - What measures were introduced to countermand these issues? How were they adopted?
  - What are the implications of the past for the future of Web security?
How to go back in time?

• Client-side code stored in The Internet Archive
  • Stores client-side code of crawled sites since 1996
  • Archives HTTP Headers (prefixed with X-Archive-Orig-)

• Analyze most important sites of the time
  • 500 most frequented domains for each year
    • *Internet Jones and the Raiders of the Lost Trackers (Lerner et al., USENIX 2016)*
    • blocked access to resources outside +/- three months from original timestamp
  • Main page + first level of same-domain links
    • 659,710 unique URLs, 1,376,429 frames, 5,440,958 scripts, 21,169,634 HTTP headers
Technologies used by the top 500 sites

![Graph showing technologies used by the top 500 sites over time]

- **JavaScript**: Steady increase from 0% to nearly 100% by 2012.
- **Flash**: Grew rapidly in the early 2000s and then declined.
- **Java**: Minimal usage, mostly stable at around 0%.
- **Silverlight**: Minimal usage, mostly stable at around 0%.

Ben Stock - USENIX 2017 - How the Web Tangled Itself
JavaScript complexity on the rise
Multiple parties contribute JavaScript code
Evolution of Client-Side Technology

Indicators of Security Awareness/Measures

Discovered Security Issues
Client-Side Cross-Site Scripting still going strong

![Graph showing domains vulnerable from 1997 to 2016.]
Insecure postMessage handling

• postMessages allow origin and destination verification
  • Protects integrity and confidentiality
Known vulnerable jQuery versions

- jQuery (all)
- jQuery (known vuln.)

CVE-2011-4969

$ params interpreted as HTML
Flash Cross-Domain Policies

- 3/4 of * wildcards on sites with potential login state
Evolution of Client-Side Technology

Indicators of Security Awareness/Measures

Discovered Security Issues
HTTP only cookies

- Introduced in 2001 for IE
  - meant as XSS mitigation
  - cookies not accessible from JavaScript

- First used in 2006, steady increase since 2009
  - almost 50% adoption in 2016
  - lower bound as crawler does not log in
Clickjacking Protection through X-Frame-Options

• Introduced in 2009 for IE/Firefox
  • ability to disallow (third-party) framing

• First used in 2010, steady increase since then
  • over 50% adoption by now

• Deprecated by CSP since 2015
  • still slight increase in 2016
HTTP Strict Transport Security

- Introduced in 2010 for Chrome/Firefox
  - auto-upgrades HTTP to HTTPS
- First used in 2012, steady increase
  - almost 30% adoption in 2016
Content Security Policy

• Introduced in 2010 for Firefox
  • explicit whitelisting of resources, e.g., scripts, images, ..

• First used in 2013, very slow increase
  • less than 10% after three years
Insights of our Analysis
Client-Side Technology

- Web’s complexity is still on the rise
  - steady increase in code size and cyclomatic complexity

- Increased involvement of third-parties
  - 12 distinct origins in 2016
  - including several vulnerable versions of libraries

- Towards a multi-origin Web
  - e.g., increase in postMessages for cross-domain communication
  - applications no longer bound to a single origin
Client-Side Security

• Client-Side XSS remains constant issue
  • up to 15% vulnerable in 2009, still around 8% in 2016

• Utility trumps Security
  • Even safe defaults are circumvented, e.g., crossdomain.xml

• Complexity of Deploying Security Measures
  • Easy to deploy measures are rolled out rapidly, e.g., X-Frame-Options
  • In contrast, CSP is very slow to market
Confirming Intuitions

- Applications become more and more complex
- Simple security mechanisms are quickly adopted
- More involved mechanisms (e.g., CSP) lack behind in adoption
- Administrators aware of general security concepts have less vulnerabilities.
Correlating Client-Side XSS and Awareness Indicators

- HTTPonly Cookies
  - Fraction of sites with HTTPonly and XSS higher than no measure and XSS

  - Early adopters rarely have an XSS, fraction increases, almost at baseline in 2016

- CSP
  - CSP sites don’t even have any Client-Side Cross-Site Scripting
  - Might be early adopter phenomenon
Threats to Validity

- Limited view into applications (missing login)
  - not all cookies stored
  - protected resources might have other headers (e.g., X-Frame-Options)

- Blocked "Bubble escapes"
  - blocked access to newer resources
  - JavaScript was collected dynamically

- However, historical results align with previous papers
  - cross-domain policies, JS inclusions, Client-Side XSS, outdated libraries
Lessons learnt from our 20-year study

• Ease of Use for Security Measures
  • simple security measures are quickly adopted

• Make Security Mandatory
  • e.g., postMessage origin must be accessed before data can be accessed
  • soft integration of stricter policies: warn first, block later

• Improve tools for and awareness of developers
  • tools help to rewrite secure code
  • updatability on libraries

```javascript
function loadAdvertisementInsecure() {
    document.write("<script src='http://ad.com/?referrer=
    + location.href + '></script>");
}
```

```javascript
function loadAdvertisementSecure() {
    var script = document.createElement("script");
    script.src = 'http://ad.com/?referrer=' + location.href;
    document.body.appendChild(script);
}
```
Conclusion

• We studied evolution of the Client side over 20 years
  • technologies used
  • discovered vulnerabilities
  • deployed mitigation techniques

• Several intuitions could be confirmed
  • However, HTTPonly cookie sites more likely to have an XSS

• Client-Side Web Security remains hard problem
  • Protection barely keeps up with increased attack surface/flaws
  • Lessons learnt from the last 20 years should be incorporated in upcoming APIs/technologies
Conclusion

• We studied evolution of the Client side over 20 years
  • technologies used
  • discovered vulnerabilities
  • deployed mitigation techniques

• Several intuitions could be confirmed
  • However, HTTPOnly cookie sites more likely to have an XSS

• Client-Side Web Security remains hard problem
  • Protection barely keeps up with increased attack surface/flaws
  • Lessons learnt from the last 20 years should be incorporated in upcoming APIs/technologies

Thanks! Questions?