AppSec Reanimated

Webinar Series

This series explores modern appsec approaches. What tools and techniques improve security within the SDLC? How do we effectively combine manual analysis with automation? How do we adopt what works and avoid what doesn't?

Out of the AppSec Abyss:

What's making modern appsec effective?

Mike Shema
mike@cobalt.io
February 8, 2017

“Another 'application' of JavaScript is to poke holes in Netscape’s security. To *anyone* using old versions of Netscape before 2.01 (including the beta versions) you can be at risk to malicious Javascript pages which can

a) nick your history

b) nick your email address

c) download malicious files into your cache *and* run them (although you need to be coerced into pressing the button)

d) examine your filetree.”

– comp.sys.acorn.misc. June 30, 1996

“No one knows who they were or what they were doing.”

– Nigel Tufnel, *Spinal Tap*

“Time to die.”

HTML5 audio/video elements (eventually) **replace** Flash.

Content Security Policy headers (partially) **mitigate** XSS.

Prepared statements (actually) **prevent** SQL injection.

Outline the Eulogy

HTML5 audio/video elements (eventually) **replace** Flash. — How can **technology** improve security?

Content Security Policy headers (partially) **mitigate** XSS. — How can **process** address legacy code?

Prepared statements (actually) **prevent** SQL injection. — How can **people** know to use available tools?

Content Security Policy directives provide revelation and restriction.

```
default-src 'self'
                default-src 'https:'
   {resource}-src 'none'
      script-src 'unsafe-eval' 'unsafe-inline'
```

How To Transfer Pages Securely

Spawn threat models
— http://

Measure risk
— https://ssllabs.com

Solve sophisticated threat models
— HSTS, HPKP

Solve fundamental threat models
— Let’s Encrypt

Let's Encrypt

Enabling widespread adoption by removing cost as initial barrier to entry.

Enabling ongoing adoption by supporting automation (ACME protocol).

https://letsencrypt.org

Have
To
Take
Problems
Seriously

From Abyss to Cloud

Continuous integration & deployment (CI/CD) reimagines separation of duties, requires increased testing.

Ephemeral systems favor retire and replace (with updated image) over patch and preserve.

Systems become peers with code; managed via API and treated as components.

From Cloud to Abyss

Avoid collapsing all security barriers between a code commit and a production system.

Avoid over-exposing secrets.

Avoid polluting test environments with production data.

Shifting Surfaces

Combining roles, striving for DevSecOps.

Virtualization, containers pushing towards syscalls instead of systems.

App stacks and libraries have ever-increasing dependency graphs, unclear security.

All Eyes Are Shallow

Bug Bounties are continuous (though unpredictable) scrutiny with coordinated disclosure.

Largely address gaps where scanners aren’t present or where scanners fail.

They’re PR — public relations and pull requests. Not security programs.

Depths of Madness

Passwords as proof of identity continue to fail as identical password are reused across applications.

Time for another PCI? — Password Control Initiative

More multi-factor instead of just more hashing.

Tokenize the password

Recovery (Facebook’s Delegated Recovery)

So is it effective?

Cloud environments favor processes that drive forward progress instead of accruing legacy systems.

Apps will always struggle with legacy code.

Continuous processes that emphasize automation and feedback loops for “normal” testing create touchpoints for security testing.

Thank you!

mike@cobalt.io

Questions?

https://webinar.cobalt.io

Stay tuned for more **AppSec Reanimated.**

And check out the **AppSec Disrupted** series, too!