Finding Bad Needles on a Worldwide Scale

Dmitry Savintsev

Yahoo!

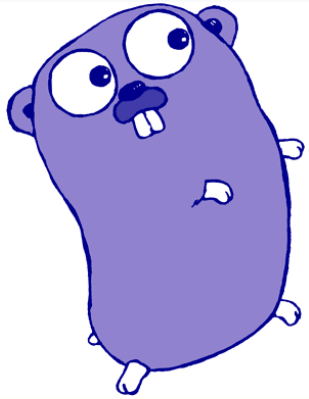
OWASP AppSecEU 15

Amsterdam, The Netherlands

Who I am

YAHOO!

- Security Engineer (Paranoid Labs)
- Developer and Paranoid
- Custodian of internal XSS scanner
- “Gopher” – Go champion



Agenda

- Scope: Reflected XSS & Big Scanning
- Webseclab – test suite and playground
- Scanmus – internal XSS scanner
- Gryffin - Scaling up through CD
- Contextdetect - Fight for Quality
- Next Steps
- Lessons and Summary

Reflected “Server-Side” XSS

- Large scale automated scanning
- Focus on a specific vulnerability type
  - reflected server-side XSS
- Improve and learn
- XSS remains a leading cause of incidents
  - and BugBounty payouts

Cross-Site Scripting (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are two different types of XSS flaws: 1) Stored and 2) Reflected, and each of these can occur on the a) Server or b) on the Client. Detection of most Server XSS flaws is fairly easy via testing or code analysis. Client XSS is very difficult to identify.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

Screenshot from https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)

Large Scale Testing

- Low tolerance for noise / False Positives
- Large websites require *high quality* automatic scanning
- Claim: **accurate** detection of reflected XSS flaws is not yet a fully solved problem!

Webseclab

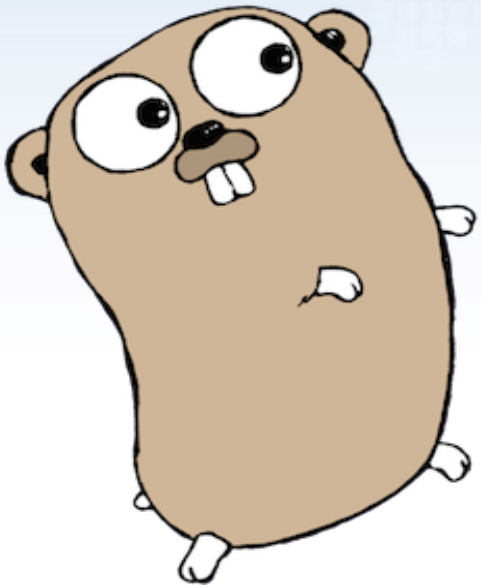
OWASP AppSecEU 15
Amsterdam, The Netherlands

Why Webseclab

- Need for “frozen” tests
  - cannot wait for the right thing to come up in prod!
  - need to model the previous cases
- “Playground” – environment to experiment and iterate
- Documentation and communication

Webseclab

- Based on multiple (internal) predecessors
  - ad-hoc PHP scripts, NodeJS app
- In Go - for ease of deployment, and more!



Webseclab Cases

- Reflected and DOM XSS
- Real-life cases
  - Collection of Yahoo XSS experience
- Includes real issues as well as False Positives (FPs)
  - learn from both kinds of scanning mistakes, False Negatives and False Positives

Easy Install

1. Download the binary from https://github.com/yahoo/webseclab/releases
2. chmod 755 webseclab-mac
3. Run it!

```
$ ./webseclab-mac
```

=> Webseclab running on http://127.0.0.1:8080!

Webseclab Demo

- http://127.0.0.1:8080

Open Source Software

- Open source:
  - http://github.com/yahoo/webseclab
- First Yahoo published Go project
- First Yahoo security project on Github

Webseclab and Open Source

- Open source:
  - http://github.com/yahoo/webseclab
- Would love open source security projects to use it (more)
  - as well as anyone else! (Training, teaching…)
- Plans to use Webseclab for OWASP ZAP CD testing

Arachni & Webseclab

- 3 issues identified – all fixed
- Segmentation fault on a webseclab test:

https://github.com/Arachni/arachni/issues/543

- Double-encoded payload (doubq.1):

https://github.com/Arachni/arachni/issues/581

- Textarea injections:

https://github.com/Arachni/arachni/issues/579

OWASP ZAP

https://code.google.com/p/zaproxy/issues/list?can=2&q=XSS&sort=-id

a few False Positive and False Negative issues identified and reported:

- XSS False Positive on injections into script block (Webseclab /xss/reflect/js3_fp?in=)
- XSS False Negative on double-encoded script injections
- XSS False Negative on script injections into the Referer HTTP header
- False Negative XSS on injection outside of HTML tags

w3af

- Many _fp URLs show in the scan results
- Will follow up with the project members
  - Possibly a feature not a bug

Industry Parallels

WAVSEP
Web Application Vulnerability Scanner Evaluation Project

https://github.com/google/firing-range

Gruyere:

Web Application Exploits and Defenses

A Codelab by Bruce Leban, Mugdha Bendre, and Parisa Tabriz

Screenshots and images taken from:
Gruyere: https://google-gruyere.appspot.com/
Wavsep: https://code.google.com/p/wavsep/
Webgoat: http://webgoat.github.io/

Scanmus

OWASP AppSecEU 15
Amsterdam, The Netherlands

Scanmus

- Yahoo internal reflected XSS scanner
  - picks up a few other issues as well (SQLi, path traversal, etc.)
- Written by Rasmus Lerdorf while at Yahoo
- Helped to find many XSS bugs
- Missed by many ex-Yahoos!

Scanmus Demo

Scanmus internals

(simplified version)

- A set of tests
  - all based on real issues and incidents
- Request payload
- Expected string or regexp
- If matches, show as a (potential) finding.

Scanmus test example

```
 'full.1' =>
array('send'=>'%22%3E%3Cscript%3Ealert(%22xss%22);%3C
%2Fscript%3E',
    'expect'=>'"><script>alert("xss");</script>',
    'fail_msg'=>'Full Javascript hack worked!',
    'notify'=>true, 'replace'=>1,
    'level'=>3,
    'name'=>'full.1',
    'charset_check'=>false,);
```

Sample payloads

- %22onmouseover=%22alert(document.cookie)
- foo%20onmouseover=alert(document.cookie)//
- javascript:alert(123);(//
- alert(142);(
- foo'+alert('xss')(//
- </script>foo<script>alert(135);</script>
- %0d%0a%0d%0a%3Cscript%3Ealert(document.cookie)%3C/script%3E
- "><img src=foobar onerror=alert(13579)>
- <img src=foobar onerror=alert(13579)>

Context detections

- Context adjustments done with regular expressions:

```
// if single_quote_closing_tag_check is set, we check to see if the injection happens inside a tag with a single-quoted attribute.  If it doesn't, then we ignore this hit.
if(isset($test['single_quote_closing_tag_check']) && $test['single_quote_closing_tag_check']) {
    $m = preg_quote($test['expect']);
    if(!preg_match("~=\s*'[^']*{$m}[^<]*>~",$text)) return;
    if(preg_match("~=\s*\"[^\"]*{$m}[^<]*>~",$text)) return;
}
```

Gaps and Problems

- Speed and coverage
  - single-threaded scanner
  - some scans taking hours or days!
- Accuracy, especially False Positives
  - overwhelming noise
- Quality of findings and reports
  - difficult to understand

Solution Direction

- ~~Rewrite everything from scratch!~~
- Set up a reliable test suite
  - Webseclab
- Separate crawler and fuzzer
- Accuracy: brainstorm on better context detection

Gryffin and CD

OWASP AppSecEU 15
Amsterdam, The Netherlands

Continuous Delivery

- Company Direction: Launch Velocity
- “Commit to production with no human intervention”
- Needed to adapt security scanning

CD “Firehose”

- Hundreds of releases **per day**
- Number of Gryffin scans per month:

Month	# of scans
01/15	6,482
02/15	19,780
03/15	43,538
04/15	13,226

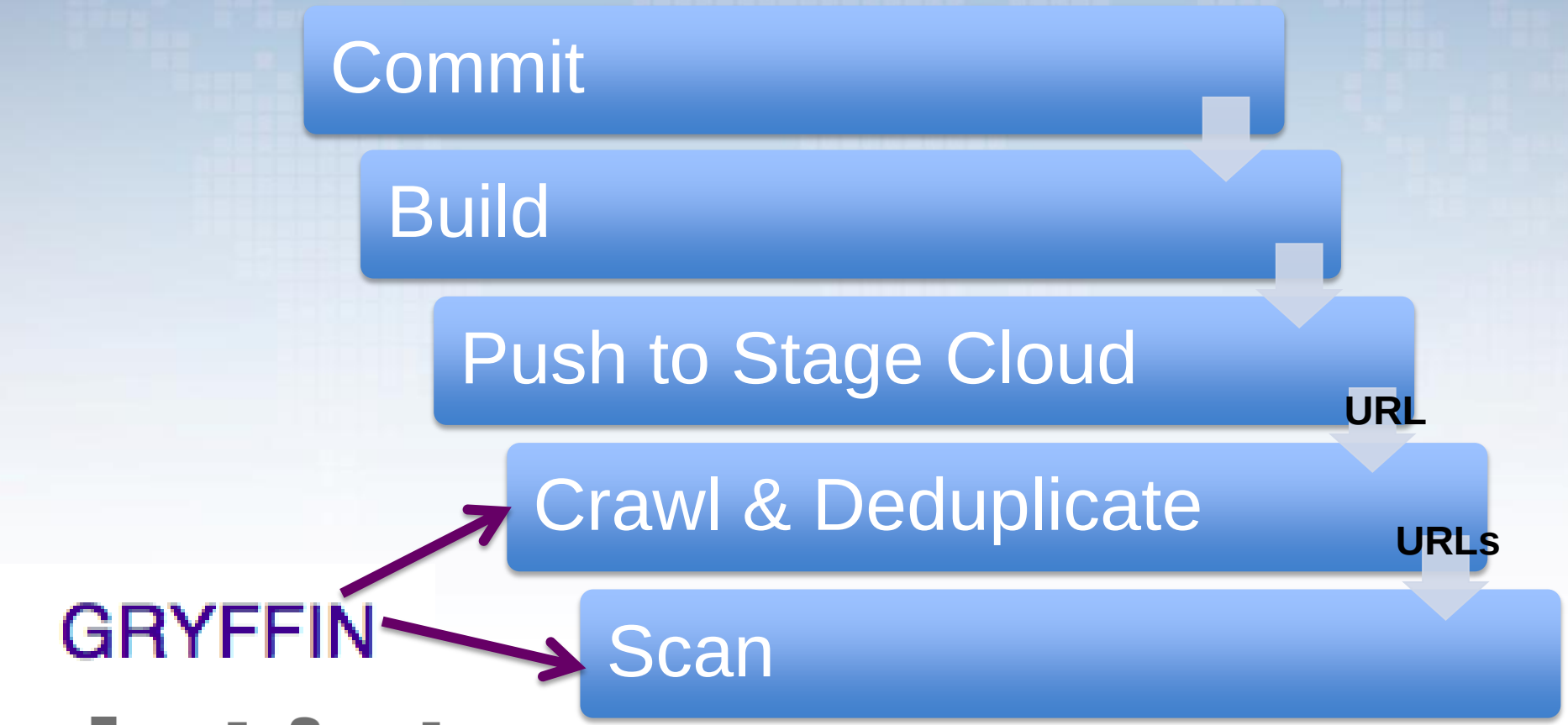
Gryffin

- Optimized crawler
  - Smart deduplication
- Framework to run multiple scanners
  - both internal and open-source
- Management of distributed tasks
- Reporting – aggregation of findings

Gryffin plugins

- Scanmus
- Tainted PhantomJS (DOM XSS scanner)
- Arachni scanner
- skipfish
- sqlmap

CD integration



Gryffin Results

- Continuous CD-driven scanning
- Less reliance on engineers doing scans
- Comparison of results from multiple scanners
  - Scanmus vs. Arachni vs. Skipfish …

Coming soon…

Gryffin talk accepted for OWASP AppSecUSA ’15!

September 22-25, 2015 in San Francisco

OWASP AppSecEU 15
Amsterdam, The Netherlands

Gaps and Problems (Gryffin)

- Accuracy of scanners
- Large amount of False Positives
  - especially noticeable when you scan (almost) everything!
- Difficult to triage and analyze
- Must do something, quick!

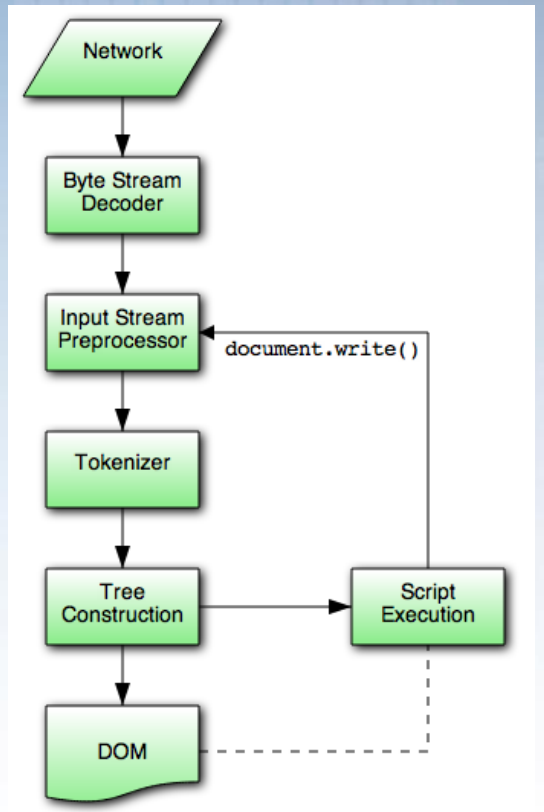
Contextdetect

Scanner and Context

- Context vital for:
  - Secure coding (prevention)
  - Scanning or testing (detection)
- Consider alert(document.cookies):
  - Innocuous in the normal HTML text context
  - Executable in Javascript block
  - … unless in a properly quoted string!

Parsing “boxes”

- HTML or JS parser break the source into “boxes”
- Inside of “box” same processing
- User input should never be able to draw its own boxes and borders between them!



Injection detection theory (1)

Active content – Javascript/CSS

- Inject a “breaker”
  - ex. Javascript with unbalanced parens
- Check if syntax is broken
  - Using a **real** parser

Injection detection theory (2)

HTML Contexts

- Inject ABC where:
- “A” and “C” are identifiable unique strings
- B is the “context breaker”
  - ex. single or double quote, a tag, a space…
- Check if A and C are in the same context “box”
  - Using an HTML5 parser

Contextdetect

- Go-based library and application
- Uses an HTML5 parser & Javascript parser

http://godoc.org/golang.org/x/net/html

http://godoc.org/github.com/robertkrimen/otto/parser

- Performs verification of Scanmus findings
- Connected via JSON bridge
  - Microservices!

Contextdetect Impact

- Allowed to achieve practically 0% False Positive rate!
- Reduced False Negatives as well
- Allowed to provide more meaningful findings messages
  - based on the context unit, not line of response

Summary / Next Steps

Next Steps

- Writing a Go-based scanner
  - Context-driven scenario-based detection
- Using experience (and confidence) from the Contextdetect project
- Using and growing Webseclab tests
  - more open-source community outreach

Lessons

- Scanner / tool is only as good as its **tests**
- Use **multiple** scanners for cross-checking and to get the best of each
- **Real parsers** (HTML5/JS/CSS) for **accurate** context-based detection and verification
- **Go** is an **effective** tool for **large-scale** server-side systems (security and more)

Summary

- Webseclab – foundation for improving scanning systems
- Gryffin – framework for scaling up, CD integration, multiple tool plug-ins
- Contextdetect – using Go HTML5 and Javascript parsers for context-oriented verification, eliminated known False Positives.

Questions?

OWASP AppSecEU 15
Amsterdam, The Netherlands

Thank You!

Dmitry Savintsev

@dimisec

https://github.com/dmitris

dsavints@yahoo-inc.com

Image attributions

Go gopher is the work of Renee French licensed under Creative Commons Attributions 3.0:
https://blog.golang.org/gopher

http://gommavulcanizzata.deviantart.com/art/Inkscape-Lightbulb-94339717?q=gallery%3AGommaVulcanizzata%2F7267989&qo=0

http://commons.wikimedia.org/wiki/File:U.S._Navy_Petty_Officer_3rd_Class_Jordan_Crouse_aims_a_fire_hose_on_a_simulated_fire_during_a_general_quarters_drill_aboard_the_amphibious_assault_ship_USS_Iwo_Jima_(LHD_7)_as_the_ship_operates_in_the_Gulf_of_Aden_1 (original: http://www.defense.gov/dodcmsshare/newsphoto/2012-10/hires_121017-N-OR551-040.jpg, Public domain photograph from Defense.gov News Photos archive)

https://www.flickr.com/photos/vialbost/12481376133/ - “Merci / Thank you”