Why Fuzzing Web Apps is Hard

Simon Bennetts @psiinon
OWASP ZAP Project Lead
OpenSSF Security Tools WG
StackHawk Inc

2021 March 24

FUZZCON EUROPE
WebSecurity edition

Whats ZAP?

- Tool for finding vulnerabilities in web apps
- Completely free and open source
- Worlds most frequently used web app scanner / DAST tool

Fuzzing Vs DAST

- Both attack running systems
- Fuzzing – lots of ‘random’ data
- DAST – highly targetted attacks
- ZAP – manual fuzzing only

Problem: Time

- Time is the enemy of DAST tools
- ZAP active scan rule strengths:
  - Low 6 reqs / param / page
  - Medium 12
  - High 24
  - Insane 100s?

Problem: Time

- Example Site: 100 pages, average 5 params per page
  - = 500 parameter targets
- 20 core active scan rules, medium strength (12 reqs)
- 500 x 20 x 12 = 120k requests
- 100 req/sec = 1,200 secs == 20 mins
- 10 req/sec = 12,000 secs == 200 mins == 3.3 hours

Problem: Time

- But then consider
  - ~ 40 beta active scan rules
  - Headers
  - Path elements
  - High attack strength

Problem: Time

- Example Site: 200 pages, 5 params + 5 headers + 4 path elements per page (14)
  - = 2,800 parameter targets
- 20 core + 40 beta scan rules, high strength (24 reqs)
- 2800 x 60 x 24 = 4,032,000 requests
- 100 req/sec = 40,320 secs == 672 mins == 11.2 hours
- 10 req/sec = 403,200 secs == 6720 mins == 112 hours
  - = 4.6 days

Solutions: Time

- Make fewer requests
- More targetted requests
- Understand app structure better
- Speed up target system
- Live with it ;)

Problem: Discoverability

- Fuzzing functions – definitions via language
- SAST – access to all of the code
- Web apps have no definition
- API definitions exist, but often not available
- Unit tests great, but often don’t exist or not complete
- Crawlers / spiders help but have limitations

Solutions: Discoverability

- API definitions
- Site maps
- Comprehensive unit tests
- Use standard HTML controls

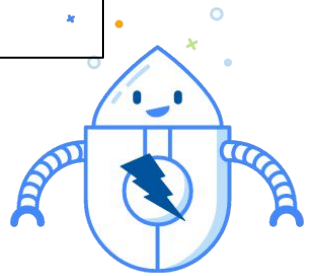
Problem: Good Test Data

- Register user form:

Register New User

User name:	test
Password:	**********
Repeat password:	**********

Register



- Error: User name must be an email address

Solutions: Good Test Data

- Comprehensive unit tests
- Manual configuration

Problem: Understanding web app structure

- DAST tools build up an internal map – ZAP Sites Tree
- App map too small: scan misses features
  - URL parameters which represent site structure ..&page=home
  - URL parameters which represent different actions ..&action=add
- App map too big: scan takes too long
  - Database driven content ../company/team/..

Solutions: Understanding web app structure

- Autodetection (hard)
- Manual configuration

Problem: Authentication

- Too many authentication mechanisms
- Really hard to tell which is in use
- How to tell if authenticated or not?
- Anti-automation features
- Single Sign On

Solutions: Authentication

- Authentication tokens
- Turn it down / off! (in test environment;)
- Autodetection (hard)
- Manual configuration

Problem: Session Handling

- If not maintained then have to continually re-authenticate
- Cookie based straight forward
- Client side tricker
- Logout links
- Other session invalidating events

Solutions: Session Handling

- Autodetection (easier)
- Manual configuration
- Use ‘standard’ mechanisms

Problem: Issue Detection

- Some issues very visible in web UI
  - Stack Traces
  - Reflected XSS
- Others trickier
  - Blind SQL
  - Out of band issues
- DAST tools typically cant see inside an app

Solutions: Issue Detection

- Mostly down to the tool
- Manual configuration
- Server side detection (e.g. log analysis)
- OAST (Out of band App Service Testing)
- IAST (Interactive App Service Testing)

Summary of the Problems

- Time
- Discoverability
- Good Test Data
- Understanding Web App Structure
- Authentication
- Session Handling
- Issue Detection

Find Out More

- ZAP
  - www.zaproxy.org
  - @zaproxy
- OpenSSF Security Tools WG
  - openssf.org