Fuzz Testing of Web Applications

Rune Hammersland & Einar Snekkenes

Outline

- The Problem
- What is fuzzing?
- What we have done
- Experiment
- Findings
- Conclusions

The Problem

- Bad handling of input in web applications leads to vulnerabilities.
- Testing input handling for web applications is hard.
- Large number of possible inputs, selection is hard.
- Letting the programmer choose test values might not be wise.

What is Fuzzing?

- Random testing technique discovered by Miller et al. "a dark and stormy night."
- Study from 1990 done on UNIX command line applications shows about a third crashed or hung.
- Later studies includes:
  - Command line: GNU/Linux, Windows NT, Mac OS X
  - GUI programs: X11, Windows NT/2000, Mac OS X

What is Fuzzing?

- Has also been used for:
  - —Month of Kernel Bugs
  - —Month of Browser Bugs
  - —Programming libraries
  - —Network protocols

We Have:

- Evaluated fuzzing for web applications (is the method feasible in this area?) by:
  - —Creating a tool chain for fuzzing web applications, and
  - —Used the tool chain to find weaknesses in web applications.

Overview of Findings

Categories of errors:

E1: Resource exhaustion

E2: Failure to handle exceptions

E3: Failure to validate input on server

E4: Non-semantic use of HTTP status codes

Experiment

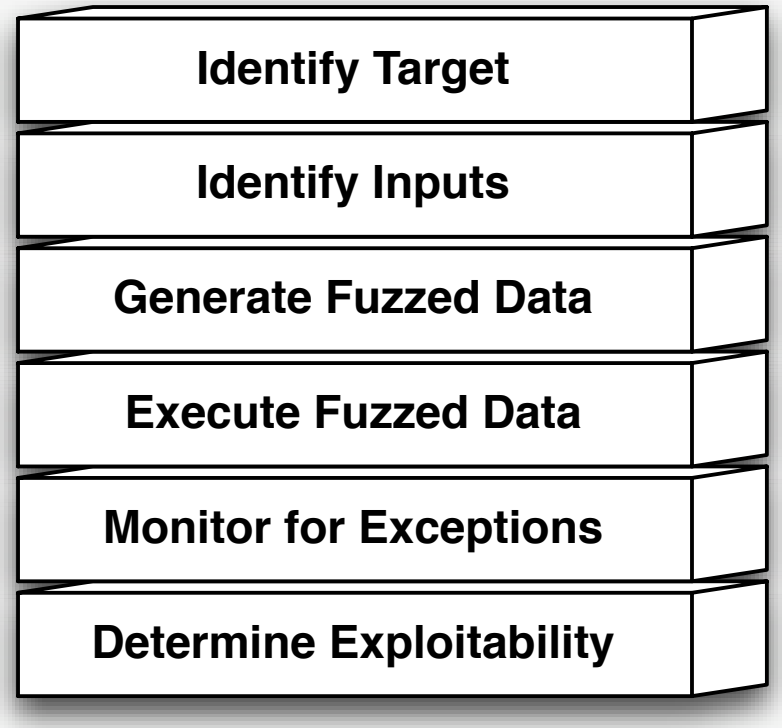


Figure taken from
"Fuzzing: Brute Force Vulnerability Discovery"
by Sutton and Amini

Experiment (targets)

- Chyrp (weblog)
- eZ Publish (CMS)
- Junebug (wiki)
- Mephisto (CMS)
- ozimodo (weblog)
- Request Tracker (ticketing system)
- Sciret (knowledge base)
- Wordpress (weblog)

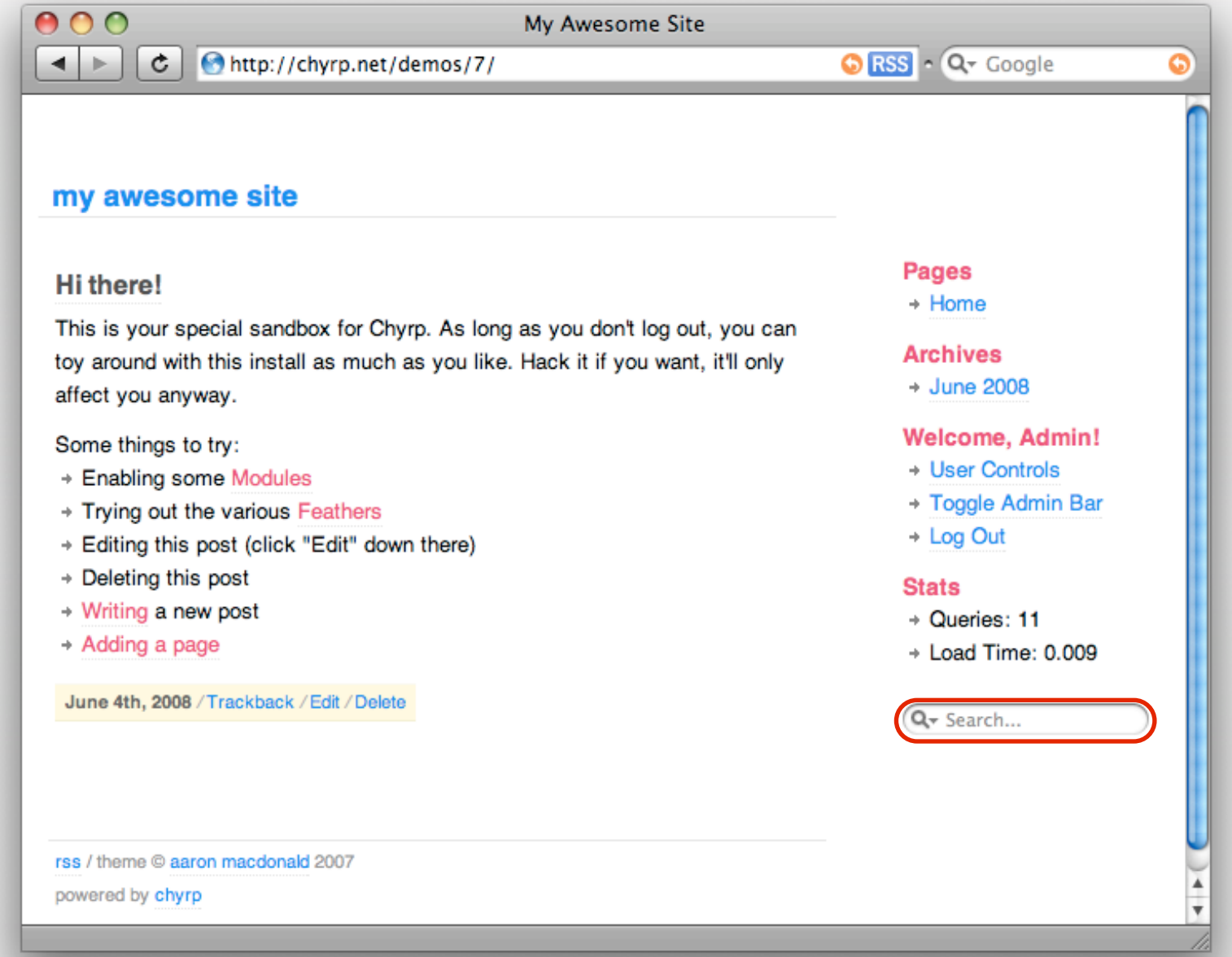
Experiment (identify input)

- We used a crawler to traverse the pages of the applications.
- An attack script template is created with information on the forms found.

Experiment (identify input)

Application	#forms	#inputs
Chyrp	4	11
eZ	6	20
Junebug	5	8
Mephisto	10	49
Ozimodo	5	26
RT	4	64
Sciret	6	24
Wordpress	4	10
Sum	44	212

Experiment (identify input)



Experiment (identify input)

```
<form action="/chyrp/" method="get">
  <input type="hidden" name="action" value="lookup" />
  <input type="search" name="query" value="Search …" />
</form>
```

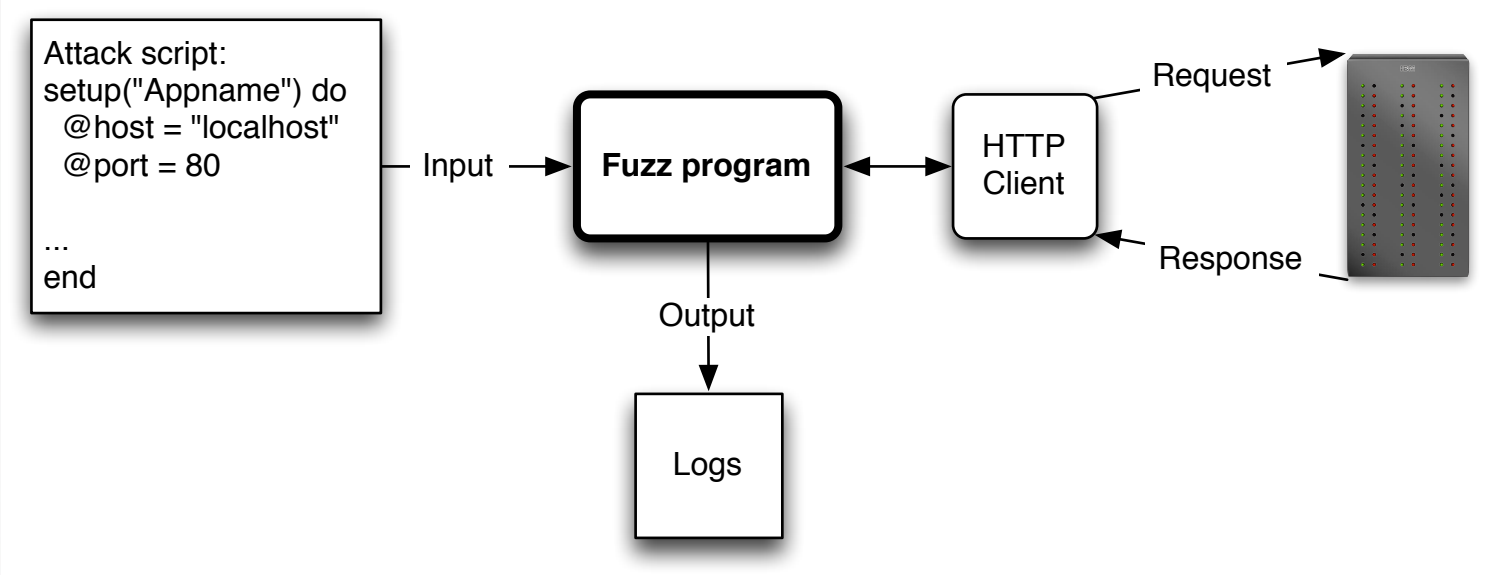
Experiment (generate fuzz)

```
attack("Search box") do
  url = "/chyrp/"
  many :get, url, {:action => "lookup", :query => str(100)}
  many :get, url, {:action => "lookup", :query => byte(100)}
  many :get, url, {:action => "lookup", :query => big}
end
```

A random request could look like:

```
GET /chyrp/?action= lookup&query=4444596086
```

Experiment (fuzzing)



- Invoke the fuzzer with the attack script.
- Useful to monitor logs from web server.
- Fuzzer dumps log files in a directory for later analysis.

Findings

Application	E1	E2	E3	E4
Chyrp	—	—	—	—
eZ	—	—	—	—
Junebug	—	—	3	—
Mephisto	—	2	1	—
Ozimodo	—	2	—	—
RT	1	—	—	—
Sciret	—	—	—	—
Wordpress	—	—	—	2
Sum	1	4	4	2

E1: Resource exhaustion
E2: Failure to handle exceptions
E3: Failure to validate input on server
E4: Non-semantic use of HTTP status codes

Conclusions

- Web applications are vulnerable to fuzzing.
- Small tests found some results.
  - —More comprehensive tests should be done for "real" testing.
- Generation of attack scripts should be automated further.

Questions?