A testbed for the evaluation of intrusion prevention systems

Jeff Stuckman
Finding security-critical software faults

• Static analysis
  – Buffer overflow detection for C programs

• Dynamic analysis
  – Penetration testing
    • Web application vulnerability scanners
  – Intrusion detection systems
    • Network traffic monitors
  – Intrusion prevention systems
    • Web application firewalls
    • Platform-specific security enhancements
Testing and evaluation of fault-detection tools

• Static analysis
  – Buffer overflow benchmark suite

• Dynamic analysis
  – Penetration testing
    • Deliberately buggy web applications
  – Intrusion detection systems
    • Network traffic replay
  – Intrusion prevention systems
    • ???
    • (ad-hoc evaluation is common in academic research)
Challenges in evaluating intrusion protection systems

• IPS only responds to a penetration in progress
• IPS could disrupt legitimate traffic
• IPS’s algorithms may learn from legitimate traffic
Evaluation of intrusion prevention systems

• True-positive and false-positive rate/ precision and recall
• Non-disruption of legitimate workloads
• Choosing vulnerabilities to test
  – Synthetic applications vs real vulnerabilities
  – Applications without vulnerabilities (control group)
  – Broad representation of vulnerability classes?
  – Representative sample of vulnerability classes from CVE (bottom-up benchmark)?
## Developing vulnerability modules

<table>
<thead>
<tr>
<th>Application with vulnerability present</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application with vulnerability patched</td>
</tr>
<tr>
<td>Application database</td>
</tr>
<tr>
<td>Application installer scripts</td>
</tr>
<tr>
<td>Attack generator</td>
</tr>
<tr>
<td>Attack validator</td>
</tr>
<tr>
<td>Background (legitimate) workload generator</td>
</tr>
<tr>
<td>Background workload validator</td>
</tr>
</tbody>
</table>
Vulnerability module concept

- SQL injection
- XSS attack
- XSRF attack
- Request validation
- Cookie tampering

Web server

Attack generator

Intrusion prevention system
Demo

• Incorporates:
  – Two applications (Mediawiki, Drupal)
  – Three vulnerabilities (XSS, XSRF, SQL injection)
  – One intrusion prevention system (Apache mod_security)
Implementation challenges
Finding test applications

- [http://www.exploit-db.com/](http://www.exploit-db.com/)
- Vulnerability reporting is often vague
- Vulnerability fixes are not immediately announced
- Some vulnerabilities are non-exploitable
- Old applications sometimes fail to install on newer platforms
- Vast majority of vulnerabilities are for PHP
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Plat.</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-05-10</td>
<td>Traidnt UP (view.php) SQL Injection Vulnerability</td>
<td>18</td>
<td>ScOrKIOOn</td>
</tr>
<tr>
<td>2011-05-09</td>
<td>Joomla Component com_hello SQL Injection Vulnerability</td>
<td>27</td>
<td>the_cyber_nuxble</td>
</tr>
<tr>
<td>2011-05-09</td>
<td>Joomla Component com_versioning SQL Vulnerability</td>
<td>30</td>
<td>the_cyber_nuxble</td>
</tr>
<tr>
<td>2011-05-04</td>
<td>Setthink HTMT Menu SQL Injection Vulnerability</td>
<td>181</td>
<td>Caddy-Dz</td>
</tr>
<tr>
<td>2011-05-03</td>
<td>Horizon Web Builder (fshow.php) SQL Injection Vulnerability</td>
<td>157</td>
<td>Iolo Morganvlg</td>
</tr>
<tr>
<td>2011-05-02</td>
<td>Travel11 SQL Injection Vulnerability</td>
<td>188</td>
<td>Caddy-Dz</td>
</tr>
<tr>
<td>2011-05-02</td>
<td>Permian OpenData CMS SQL Injection Vulnerability</td>
<td>147</td>
<td>Alexander</td>
</tr>
<tr>
<td>2011-04-29</td>
<td>SOOP Portal Raven 1.0b SQL Injection Vulnerability</td>
<td>174</td>
<td>Evil-Thinker</td>
</tr>
<tr>
<td>2011-04-27</td>
<td>Symphony CMS 2.1.2 Blind SQL Injection</td>
<td>228</td>
<td>Wireghead</td>
</tr>
<tr>
<td>2011-04-26</td>
<td>WordPress SessionWidner Plugin 0.43 SQL Injection</td>
<td>314</td>
<td>Ma33Tr2-Dz</td>
</tr>
<tr>
<td>2011-04-25</td>
<td>mySealIT 0.1781 SQL Injection Vulnerability</td>
<td>176</td>
<td>AutoSec Tools</td>
</tr>
<tr>
<td>2011-04-24</td>
<td>SoftMP3 SQL Injection Vulnerability</td>
<td>244</td>
<td>mAvTi</td>
</tr>
<tr>
<td>2011-04-22</td>
<td>Realmarketing CMS Multiple SQL Injection Vulnerabilities</td>
<td>267</td>
<td>Xecut0N3r</td>
</tr>
<tr>
<td>2011-04-22</td>
<td>WebZProject 2.3 SQL Injection Vulnerability</td>
<td>255</td>
<td>AutoSec Tools</td>
</tr>
<tr>
<td>2011-04-22</td>
<td>First Escort Marketing CMS Multiple SQL Injection Vulnerabilities</td>
<td>217</td>
<td>NoNameMT</td>
</tr>
<tr>
<td>2011-04-20</td>
<td>Ultimate eShop Error Based SQL Injection Vulnerability</td>
<td>301</td>
<td>Remos</td>
</tr>
<tr>
<td>2011-04-17</td>
<td>osPHPSite SQL Injection Vulnerability</td>
<td>395</td>
<td>virres</td>
</tr>
<tr>
<td>2011-04-16</td>
<td>Shape Web Solutions CMS SQL Injection Vulnerability</td>
<td>297</td>
<td>Ashlyane Digital</td>
</tr>
<tr>
<td>2011-04-16</td>
<td>Bedder CMS Blind SQL Injection Vulnerability</td>
<td>260</td>
<td>Xecut0N3r</td>
</tr>
<tr>
<td>2011-04-16</td>
<td>Blue Hat Sensitive Database Disclosure Vulnerability SQL</td>
<td>249</td>
<td>Xecut0N3r</td>
</tr>
</tbody>
</table>
Generating attack traffic

• Complex sequences of HTTP requests are often necessary
• Exploits can be targeted with a variety of payloads
• Application upgrades sometimes change the sequence of requests necessary to operate the application
Validating attacks

• Intrusion prevention systems may rewrite offending XSS or SQL strings to neutralize them
  – Should a neutralized injection still be considered an attack?

• Full parsing of HTML or SQL strings is ultimately necessary
Generating background workloads

• Application upgrades can disrupt workload generation
• Users can perform a variety of functions
• User activity is not uniformly distributed
Conclusions

• Past research has not developed an effective solution for evaluating intrusion prevention systems
• The simulation and validation of user and attacker activity is the biggest challenge for a testbed
  – The testbed should provide common libraries of workload generators, attack payloads, and attack validators
• Assembling a vulnerability corpus is easier, but the distribution of the corpus affects the validity of the corpus as a benchmark