Intrusion Detection and Malware Analysis

Malware collection

Pavel Laskov
Wilhelm Schickard Institute for Computer Science
Motivation for malware collection

- Understanding vulnerabilities and attack techniques
- Development of protection and neutralization tools
- Understanding the attacker communities and their “business models”.
Malware collection tools

- **Honeypot**: an isolated, unprotected and monitored system, containing seemingly valuable for attacker resources, aimed at collecting examples of malicious activity.

- **Honeyclient**: an automated client-side vulnerable system executed in a controlled environment.

- **Honeynet**: a distributed collection of honeypots and email filters intended for a large-scale collection and observation of malware.
Honeypot taxonomy

- **Low-interaction**: simple daemons simulating network services; no exploitation.
- **Medium-interaction**: emulated vulnerabilities for attracting and executing malware in a controlled environment.
- **High-interaction**: real systems communicating with malware in a controlled environment.
Low-interaction honeypots

(+ ) Low security risks due to emulation
(+ ) Simple installation and recovery
(+ ) Suitable for analysis of automatic attacks
(+ ) High scalability
(− ) Not suitable for detection of interactive attacks due to limited emulated functionality
(− ) Hardly suitable for acquisition of malware binaries
High-interaction honeypots

(+) Suitable for detection and acquisition of any malware kinds
(−) Time and resource consuming installation and maintenance
(−) High security risks: additional security mechanisms are required
(−) Virtualization can be detected by malware
Medium-interaction honeypots

(+ ) A relatively wide exploit coverage
(+ ) Extensive monitoring and collection functionality
(+ ) Full virtualization not necessary
(+ ) Relative ease of deployment and maintenance
(+ ) Low to moderate security risks (egress outbreak)
(− ) Manual emulation of vulnerabilities still necessary
(− ) Detection of novel exploits not always reliable
A honeypot example: Nepenthes

- **Vulnerability modules**: emulate vulnerable parts of network services.
- **Shellcode parsers**: analyse shellcode to locate its source.
- **Fetch modules**: download binaries from remote locations.
- **Submission modules**: store binaries in a specified location.
Nepenthes vulnerability modules

Poor man’s implementation of the original vulnerability

- Send $N$ fixed strings, random junk, exploit “stages”
- Dismiss intermediate received stages
- Record final stage and use in payload

Example:

```c
ConsumeLevel LSASSDialogue::incomingData(Message *msg) {
    m_buffer->add(msg->getMsg(), msg->getSize());

    char reply[512];
    for (int32_t i = 0; i < 512; i++) {
        reply[i] = rand() % 256;
    }
}
```
Nepenthes shellcode analysis

- Analyze the incoming payload and extract malware location
- shellcode-signature module
  - Signature-controlled shellcode analyzer
  - Perl-compatible RE patterns for commonly seen shellcode
  - Identify parameters of shellcode (ports, URIs, ...)
- Shell emulator with arbitrary commands
Download the actual malware from previously generated URL

Several modules for various protocol:
- HTTP(S), (T)FTP, RCP, ...
- “Proprietary” malware protocols:
  - CSend and CReceive from AgoBot
  - LinkBind and LinkConnectback from linkbot

RFC-incompliant implementations of HTTP and FTP
### Objective:
Detection of attacks directed at client-side software, mostly web browsers:
- browser exploits
- “drive-by downloads”
- typo-squatting

### Applications:
- security analysis of web sites
- finding malicious content distribution sites
- detection of new browser exploits
- malware collection
Browser exploitation via redirection

1. Obfuscated Java Scripts
2. Third-Party Redirection
3. Malicious Scripts attempting to exploit Multiple vulnerabilities
4. Malware Installation

Source: Yi-Min Wang, Microsoft Research
Honeyclient example: HoneyMonkey

- A VM-based high interaction honeyclient, running a vulnerable browser.
- Automatic detection of redirection relationships between content distribution sites
- Detection of zero-day attacks
HoneyMonkey architecture

Stage 1: \( N \) URLs per VM, unpatched WinXP, no redirection analysis

Stage 2: 1 URL per VM, unpatched WinXP SP2, redirection analysis

Stage 3: 1 URL per VM, patched WinXP SP2, redirection analysis

“Interesting” URLs

Exploit URLs

Analysis of exploit URL density

Exploit URL topology graph

Access blocking

Zero-day exploits

Vulnerability patching

Zero-day exploits
HoneyMonkey deployment results

Results were obtained in May-June 2005 on a list of 16,190 URLs with known bad content (pornography, adware distribution, some shopping and freeware screensaver sites).

<table>
<thead>
<tr>
<th>HoneyMonkey configuration</th>
<th>Exploit num./freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1, fully unpatched</td>
<td>207 (1.3%)</td>
</tr>
<tr>
<td>Stage 2, fully unpatched (SP1)</td>
<td>688 (4.2%)</td>
</tr>
<tr>
<td>Stage 2, fully unpatched (SP2)</td>
<td>204 (1.3%)</td>
</tr>
<tr>
<td>Stage 3, SP2 partially patched</td>
<td>17 (0.1%)</td>
</tr>
<tr>
<td>Stage 3, SP2 fully patched</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>

In July 2005, 27 URLs were discovered that distributed a zero-day exploit.
Goals:
- Wide coverage of up-to-date “malware landscape”
- Fast discovery new malware strains

Challenges:
- Maintenance: deployment by less qualified administrators
- Security: avoid potential infection of host systems
- Automation: adjust to potentially unknown vulnerabilities
- Scalability: infrastructure for storing massive amounts of malware
- Utility: interface for analysis tools
- Stealth: should not be detectable by malware
Honeynet example: SGNET

- Message extraction from TCP flows
- Generation and refinement of a finite state machine model for a communication protocol used by malware
- Generation of a honeyd-compatible script for implementation of a finite-state machine.
- Communication interface for interaction with the repository and analysis components.
Lessons learned

- Malware collection is a crucial prerequisite for understanding new malware threats and development of appropriate protection tools.
- The main difficulty of malware collection lies in having to deal with highly dynamic and heterogeneous exploitation techniques.
- Special attention has to be paid to stealthy operation and security features for malware collection.
Recommended reading

Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix C. Freiling.  
The Nepenthes platform: An efficient approach to collect malware.  

Corrado Leita, Marc Dacier, and Frédéric Massicotte.  
Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots.  

Niels Provos and Thorsten Holz.  
*Virtual Honeypots: From Botnet Tracking to Intrusion Detection*.  

Yi-Ming Wang, Doug Beck, Xuxuan Jiang, and Roussi Roussev.  
Automated web patrol with Strider HoneyMonkeys: Finding web sites that exploit browser vulnerabilities.  