Vulnerability Discovery with Attack Injection

IEEE Transactions on Software Engineering (2010)
Joa˜o Antunes, Nuno Neves, Miguel Correia, Paulo Verissimo, and Rui Neves

Park, Ji Hun
2010.08.17
Contents

- Introduction
- Attack injection tool
- Case study
- Related work
- Conclusion
- Discussion
Vulnerability

- Faults caused by wrong design, implementation mistakes, which is easy to being exploited by an attack
  - Attack means that malicious input performs some unintended and usually illegal activity
Motivation

- Software dependability is getting more importance over the years especially in networked computer systems
- Network-connected servers should consider about new threats and forms of attacks
  - Sustain long periods of uninterrupted operation
  - Corruption on the server may threaten client’s privacy

Goal of this paper

- Suggest an attack injection tool (AJECT) for the automatic discovery of vulnerabilities in software
Overview of tool AJECT

**Attack generation:**
Automatically generate attacks in 4 ways

**Injection phase:**
Execute previously generated test cases (attacks)

**Input:**
- Protocol specification

**Monitoring:**
- Monitor target system’s state while executing attacks in 3 ways
Protocol as an input

- Even though the source code of server system is not available, protocols tend to be well defined.
- AJECT offers a graphical user interface tool to input specification of protocol.
- The tool attacks servers by transmitting erroneous packets with the protocol.
Network server protocols

- 2 email protocols
  - Fully developed and commonly used

- POP protocol (POP3)
  - Post Office Protocol
  - Three states
  - Interaction through text strings

- IMAP protocol (IMAP4Rev1)
  - Internet Message Access Protocol
  - Much more complicated than POP
    - Wider functionality (ex. Create mailbox)
  - Interaction through text strings
**Attack generation phase**

- **Delimiter test**
  - Put illegal or missing delimiters
    - Double quotes as illegal delimiters
    - Omitting delimiter like ‘space’ character

- **Syntax test**
  - Violate syntax
    - Addition, elimination, reordering of each field
Attack generation phase (cont’d)

- Value test
  - Replace message specification fields with malicious tokens

**Message specification**

<table>
<thead>
<tr>
<th>Field A : Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field B : 4Digit Number</td>
</tr>
<tr>
<td>Field C : Words</td>
</tr>
<tr>
<td>Field D : Words</td>
</tr>
</tbody>
</table>

- Boundary values, some illegal values
- Illegal tokens - large strings, strange characters, collected from the exploit community

**Example**

- AUTHENTICATE <A x 1296>
- < %s x 10 >;
Attack injection tool (6/8)

- Attack generation phase (cont’d)
  - Privileged access violation test
    - Determine if the server allows unauthorized accesses
      - Specialization of the value test
    - If server allows these attacks, revealing private information or granting access to some files could be possible

Example
- CREATE / <A x 10>
- SELECT .../other-user/inbox
- SELECT "{localhost/user=\}"
Injection campaign phase

- Attack injector and three kind of monitors

Injector

AUTHENTICATE <A x 1296> < %s x 10 >;
CREATE / <A x 10>
SELECT ./../other-user/inbox
SELECT "{localhost/user="۔”

Remote Monitor

Network connection

OS

Based on UNIX or Windows

Target Server

POP or IMAP protocol

Deep Monitor

Process Manager

Shallow Monitor
Injection campaign phase (cont’d)

- Monitor

<table>
<thead>
<tr>
<th>Monitor</th>
<th>Property</th>
<th>Execute in</th>
<th>Access Target Server’s OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Monitor</td>
<td>Trace server’s main process and children in great detail</td>
<td>Target server machine</td>
<td>O</td>
</tr>
<tr>
<td>Shallow Monitor</td>
<td>Monitor server’s termination code</td>
<td>Target server machine</td>
<td>O</td>
</tr>
<tr>
<td>Remote Monitor</td>
<td>Infer server state by network connection</td>
<td>Injector machine</td>
<td>X</td>
</tr>
</tbody>
</table>
Experimental results

- Target network servers

- Total up-to-date 16 servers
- Every servers support POP and IMAP both
- 8 Window servers
- 3 Unix servers
- 5 U/W servers
- 12 commercial servers
- 4 open source servers
- 3 D/S/R monitors
- 13 Remote monitors

<table>
<thead>
<tr>
<th>E-mail Servers (POP3/IMAP4)</th>
<th>OS</th>
<th>Version</th>
<th>Build Date</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>602LAN Suite (602 Software)</td>
<td>W</td>
<td>5.0.08.0403</td>
<td>4/8/2008</td>
<td>R</td>
</tr>
<tr>
<td>Citadel*</td>
<td>U</td>
<td>7.32</td>
<td>2/17/2008</td>
<td>R</td>
</tr>
<tr>
<td>dovecot*</td>
<td>U</td>
<td>1.1.rc3</td>
<td>3/9/2008</td>
<td>D/S/R</td>
</tr>
<tr>
<td>Hexamail Server Corporate</td>
<td>U/W</td>
<td>3.1.0.002</td>
<td>-</td>
<td>R</td>
</tr>
<tr>
<td>hMailServer</td>
<td>W</td>
<td>4.4.1</td>
<td>3/9/2008</td>
<td>R</td>
</tr>
<tr>
<td>Kerio MailServer (Kerio Tech.)</td>
<td>U/W</td>
<td>6.5.1</td>
<td>5/12/2008</td>
<td>R</td>
</tr>
<tr>
<td>Mailtraq (Fastraq)</td>
<td>W</td>
<td>2.12.1.2364</td>
<td>5/8/2008</td>
<td>R</td>
</tr>
<tr>
<td>Mdaemon (Alt-N Technologies)</td>
<td>W</td>
<td>9.6.5</td>
<td>6/19/2007</td>
<td>R</td>
</tr>
<tr>
<td>Merak Mail Server (IceWarp)</td>
<td>U/W</td>
<td>9.1.0</td>
<td>9/17/2007</td>
<td>R</td>
</tr>
<tr>
<td>NoticeWare Email Server NG</td>
<td>W</td>
<td>4.6.2</td>
<td>4/3/2008</td>
<td>R</td>
</tr>
<tr>
<td>Softalk Mail Server Corp.</td>
<td>W</td>
<td>8.5.1.431</td>
<td>10/30/2007</td>
<td>R</td>
</tr>
<tr>
<td>SurgeMail Mail Server (NetWin)</td>
<td>U/W</td>
<td>3.9e</td>
<td>4/10/2008</td>
<td>R</td>
</tr>
<tr>
<td>WinGate Email Server (Qbik)</td>
<td>W</td>
<td>6.2.2</td>
<td>7/12/2008</td>
<td>R</td>
</tr>
<tr>
<td>xmail*</td>
<td>U/W</td>
<td>1.25</td>
<td>1/3/2008</td>
<td>D/S/R</td>
</tr>
</tbody>
</table>

* - Open source;    U - Unix/Linux;    W - Windows
D - Deep monitor;    S - Shallow Monitor;    R - Remote Monitor
Experimental results (cont’d)

- Results
  - AJECT found vulnerabilities in five servers out of 16 servers
    - AJECT can be very useful in discovering vulnerabilities
  - Characteristic of vulnerable servers
    - All vulnerabilities are found with IMAP protocol
      - Indicate more complex protocols lead to more error-prone
    - All of vulnerable servers are closed source commercial servers
      - Larger and active community could make fewer flaws

<table>
<thead>
<tr>
<th>Vulnerable Server</th>
<th>Attack</th>
<th>Observable Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>hMailServer(IMAP)</td>
<td>&gt;20k CREATE and RENAME messages</td>
<td>Server becomes unresponsive until it crashes</td>
</tr>
<tr>
<td>NoticeWare(IMAP)</td>
<td>&gt;40 A01 LOGIN Ax5000 password</td>
<td>Server crashes</td>
</tr>
<tr>
<td>Softtalk(IMAP)</td>
<td>&gt;3k A01 APPEND messages</td>
<td>Server crashes after low memory</td>
</tr>
<tr>
<td>SurgeMail(IMAP)</td>
<td>A01 APPEND Ax5000</td>
<td>Server crashes</td>
</tr>
<tr>
<td>WinGate(IMAP)</td>
<td>A01 LIST Ax1000 *</td>
<td>Server deny all connections</td>
</tr>
</tbody>
</table>
Related works


❖ Fault injection
  ▪ Inject faults to study behavior in the presence of faults
    • Because of simplicity of faults, it is difficult to apply complex faults, like vulnerabilities of network servers

❖ Fuzzers
  ▪ Inject random sample inputs to discover vulnerability
    • Since it uses random samples, test cases are either too simplistic or specialized
    • Lack of monitoring mechanisms

❖ Vulnerability scanners
  ▪ Inject faults from database of known vulnerabilities
    • Unable to uncover unknown vulnerabilities
Conclusion

❖ Contribution
  ▪ Suggest automated methodology to discover vulnerabilities in software
  ▪ Suggest attack injection mechanism which can be applied without target’s source code

❖ Future work
  ▪ Experiment with other protocols
Discussion

❖ Pros
  ▪ The tool uses only well-defined part(protocols) of the program which could be poorly documented
  ▪ Well-organized experiments and excellent results

❖ Cons
  ▪ Simple idea, similar with existing test case generation and authors just added automatic monitoring
  ▪ In attack generation phase, it also needs pre-defined or collected malicious tokens
Thank You

Q & A
About authors

❖ Joao Antunes
  ◆ In PhD degree, University of Lisboa in Portugal

❖ Paulo Verissimo
  ◆ Professor at the University of Lisboa in Portugal
  ◆ Past associate editor IEEE Transaction on Dependable and Secure Computing
  ◆ Past chair of IEEE Technical Committee on Fault-Tolerant Computing
  ◆ Steering Committee of the DSN conference
  ◆ Fellow of the IEEE and the ACM