Polyglot: Automatic Extraction of Protocol Message Format using Dynamic Binary Analysis

Juan Caballero, Heng Yin, Zhenkai Liang
Carnegie Mellon University

Dawn Song
Carnegie Mellon University & UC Berkeley

October 2007
Protocol reverse-engineering

• Process of extracting the application-level protocol used by a program, without access to the protocol specification

• Encompasses extracting:
  – the Protocol Message Format
  – the Protocol State Machine

• Challenges:
  – No protocol specification
    » Many closed application protocols
  – Automatic
    » Samba, ICQ, Yahoo Messenger took years

This paper is about the Protocol Message Format
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Automatic message format extraction

- Extract the Protocol Message Format, one message at a time
- Problem: Automatically extract the message format from an unknown message given a program binary implementing the protocol
- Message Format = field sequence
  - Field: smallest contiguous sequence of application data with meaning

Use dynamic binary analysis to extract message format
Message Format Applications

• Extracting the message format is important for multiple problems:
  – Protocol analyzers
  – Application dialog replay
  – Intrusion detection
  – Fingerprint generation
  – Detecting services running on non-standard ports
Challenges

Main challenge is to identify the field boundaries:
1. Variable-length fields **boundaries**
   - Extract the direction fields (e.g. length fields)
   - Extract the separators
2. Fixed-length fields **boundaries**
   - Avoid concatenating two consecutive fields
   - Avoid splitting a single field into multiple ones

Another important challenge is to:
3. Find the keywords present in the message
   - Allow to map traffic to the protocol

We propose techniques to address these challenges
Related work

• **Previous work focuses on** network traces
• **Fundamental limitation:** lack of protocol semantics
• **Protocol Message Format:** [Cui06, Cui07, Leita05, Leita06]
  – Need complex heuristics to identify direction fields
  – Need assumptions about separator values & data encoding
  – Cannot identify consecutive fixed-length fields
• **Keywords / Application Signatures:** [Haffner05, Ma06, Cui07]
  – Need tokens to appear at same position across messages
  – Problems: variable-length fields, floating fields
• **Some previous work using binary analysis:** [Lim06]
  – Assumes knowledge of program’s output functions
Approach

- **Shadowing**: Monitor how a binary implementing the protocol processes a received message
  - How the message is parsed
  - How the message is used

```
1 0 0 1 1 0 1 0 1
?  
```

```
Fixed | Fixed | Length | Variable
```
Polyglot: system architecture

- Input: program binary and received message
- Output: message format
- Two phases:
  - Execution Monitoring → generate execution trace
  - Analysis → extract the message format
Execution monitoring phase

- Run the binary inside an emulator
- Implements dynamic taint analysis [Chow04,Suh04,Costa05, Newsome05, Crandall06]
  - Received network data marked as interesting
  - Marking propagates as data is used
  - Marking represents offsets in input stream
- Outputs an execution trace which contains
  - Executed instructions
  - Operand content during execution
  - Taint information
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Direction fields

• Direction Fields: store information about the location of another target field in the message
• Target field has variable-length
• Three types of direction fields
  – Length fields
    » Encode length of target field
  – Pointer fields
    » Encode displacement relative to another position
  – Counter fields
    » Encode position of field in a list of items
Direction field extraction

- Detect direction fields when used to increment the value of a pointer
- Two methods to increment the pointer:
  - Direct: increment computed from field and added to pointer
    » Indirect memory access that 1) accesses a tainted memory position, and 2) where the destination address has been computed from tainted data
  - Indirect: increment computed using a loop
    » Need to find loops and check if they have tainted conditions
Direction field extraction

• No assumptions about field encoding used
  » number of bytes/words/quadwords,
  » integer / string

• Variable-length fields:
  – Direction fields need to appear before variable-length fields
  – Variable-length field: the sequence of bytes between the last position belonging to the direction field and the end of the target field
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Separators

- Used by protocols to mark the boundary of variable-length fields
- Separator = Constant + Scope
  - Constant marks the boundary
  - Scope contains list of positions in the message where the separator is used
- Two separator types depending on scope: Field, In-fields
Separator extraction

- Program compares the separator value against all positions in the separator scope until it finds it.

- Intuition: find tokens that are compared against consecutive positions in the received data.

- Three step process:
  1. Summarize tainted comparisons (comparison table)
  2. Extract byte-long separators
  3. Extend byte-long separators into multi-byte separators.

### Comparison table

<table>
<thead>
<tr>
<th>Offset Positions</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0a (\n)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>0x47 (G)</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45 (E)</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x54 (T)</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2f (/)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

No assumptions about separator value or encoding.
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Keywords

• **Protocol constants that:**
  – appear in the received application data
  – are supported by the implementation

• **Keywords can be strings or numbers**
  – “Content-type” in HTTP
  – Version number

• **Useful to match traffic to protocols**
Keyword extraction

- Program compares the supported keywords against the received data
- **Intuition:** analyze true comparisons that include tainted data
- **Two phases**
  - Populate the comparison table
  - Extract true comparisons at consecutive positions
    » Break keywords at false comparisons or separators
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Multi-byte fixed length fields

- Intuition: program operates on the entire field
  - All field bytes need to be considered for comparisons, arithmetic operations…
- Most fields are independent
  - Exceptions: direction fields, checksums
- Detection: for each instruction extract the list of positions it operates on
  - If not direction field, then create a new multi-byte fixed field
- Limitation: can only extract fields up to the system’s word size → 4 bytes in 32-bit architectures
  - But, still better than previous work that separates each byte
Outline

• Introduction

• Techniques
  – Direction field extraction
  – Separator extraction
  – Keyword extraction
  – Multi-byte fixed length fields

• Evaluation
Evaluation overview

- Evaluated on 11 programs and 5 protocols: HTTP, DNS, IRC, ICQ, Samba
  - Clients & Servers
  - Windows & Linux
  - Text & Binary
- Results compared to Wireshark

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>Type</th>
<th>Size</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>2.2.4</td>
<td>HTTP server</td>
<td>4,344kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Miniweb</td>
<td>0.8.1</td>
<td>HTTP server</td>
<td>528kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Savant</td>
<td>3.1</td>
<td>HTTP server</td>
<td>280kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Bind</td>
<td>9.3.4</td>
<td>DNS server</td>
<td>224kB</td>
<td>Win.</td>
</tr>
<tr>
<td>MaraDNS</td>
<td>1.2.12.4</td>
<td>DNS server</td>
<td>164kB</td>
<td>Win.</td>
</tr>
<tr>
<td>SimpleDNS</td>
<td>4.00.06</td>
<td>DNS server</td>
<td>432kB</td>
<td>Win.</td>
</tr>
<tr>
<td>TinyICQ</td>
<td>1.2</td>
<td>ICQ client</td>
<td>11kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Beware ircd</td>
<td>1.5.7</td>
<td>IRC server</td>
<td>148kB</td>
<td>Win.</td>
</tr>
<tr>
<td>JoinMe</td>
<td>1.41</td>
<td>IRC server</td>
<td>365kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Unreal IRCd</td>
<td>3.2.6</td>
<td>IRC server</td>
<td>760kB</td>
<td>Win.</td>
</tr>
<tr>
<td>Sambad</td>
<td>3.0.24</td>
<td>Samba server</td>
<td>3,580kB</td>
<td>Linux</td>
</tr>
</tbody>
</table>

- Only Samba, HTTP results presented for brevity
- **Negotiate Protocol request**
## HTTP separators

<table>
<thead>
<tr>
<th>Separator</th>
<th>Apache</th>
<th>Savant</th>
<th>Miniweb</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>0x0d0a</code> (<code>\r\n</code>)</td>
<td>Field</td>
<td>Field</td>
<td>Field</td>
</tr>
<tr>
<td><code>0x2f</code> (<code>/</code>)</td>
<td>In-field</td>
<td>In-field</td>
<td>-</td>
</tr>
<tr>
<td><code>0x2e</code> (<code>.</code>)</td>
<td>In-field</td>
<td>In-field</td>
<td>In-field</td>
</tr>
<tr>
<td><code>0x20</code> (space)</td>
<td>-</td>
<td>In-field</td>
<td>In-field</td>
</tr>
<tr>
<td><code>0x3a20</code> (<code>: </code>)</td>
<td>In-field</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

GET /index.html HTTP/1.1
Host: 10.0.0.1
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.1) Gecko/20061208 Firefox/2.0.0.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: Keep-Alive
### HTTP keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Apache</th>
<th>Savant</th>
<th>Miniweb</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Host</td>
<td>Yes</td>
<td>NS</td>
<td>Yes</td>
</tr>
<tr>
<td>User-Agent</td>
<td>NS</td>
<td>Yes</td>
<td>NS</td>
</tr>
<tr>
<td>Accept</td>
<td>Yes</td>
<td>Yes</td>
<td>NS</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>Accept</td>
<td>Yes</td>
<td>NS</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>Accept-</td>
<td>Accept-</td>
<td>NS</td>
</tr>
<tr>
<td>Accept-Charset</td>
<td>Accept-</td>
<td>Accept-</td>
<td>NS</td>
</tr>
<tr>
<td>Keep-Alive</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>Connection</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Server**

<table>
<thead>
<tr>
<th>Add. keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
</tr>
<tr>
<td>‘HTTP/’, ‘e’, ‘Keep-Alive’</td>
</tr>
<tr>
<td>Savant</td>
</tr>
<tr>
<td>‘HTTP/1.’, ‘Keep-Alive’, ‘:’</td>
</tr>
<tr>
<td>Miniweb</td>
</tr>
<tr>
<td>-</td>
</tr>
</tbody>
</table>

GET /index.html HTTP/1.1
Host: 10.0.0.1
User-Agent: Mozilla/5.0 […]
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us, en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8;q=0.7, *;q=0.7
Keep-Alive: 300
Connection: Keep-Alive
Conclusion

• Proposed a new approach for extracting the protocol message format using dynamic binary analysis

• Proposed new techniques for:
  1. detecting direction fields
  2. detecting separators
  3. detecting multi-byte fixed-length fields
  4. extracting protocol keywords

• Evaluated techniques on 5 protocols and 11 programs. Results compared to Wireshark
Questions?