Stephan Kleber, Henning Kopp, Frank Kargl

Institute of Distributed Systems

August 13, 2018

NEMESYS
Network Message Syntax Reverse Engineering by Analysis of the Intrinsic Structure of Individual Messages
## Static Traffic Analysis

<table>
<thead>
<tr>
<th>Data (48 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data: d9000afa000000000010290000000000000000000000000000</td>
</tr>
<tr>
<td>[Length: 48]</td>
</tr>
</tbody>
</table>

| 00000 | 00 | 0c | 41 | 82 | b2 | 53 | 00 | d0 | 59 | 6c | 40 | 4e | 08 | 00 | 45 | 00 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0010  | 00 | 4c | 0a | 4f | 00 | 00 | 80 | 11 | cc | 40 | c0 | a8 | 32 | 32 | 42 | 6f |
| 0020  | 2e | c8 | 00 | 7b | 00 | 7b | 00 | 38 | be | d5 | d9 | 00 | 0a | fa | 00 | 00 |
| 0030  | 00 | 00 | 00 | 01 | 02 | 90 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0040  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0050  | 00 | 00 | c5 | 02 | 04 | ec | ee | d3 | 3c | 52 |
Static Traffic Analysis

Observable in Transit
Requires no access to program of entities
Static Traffic Analysis

Observable in Transit

Requires no access to program of entities

<table>
<thead>
<tr>
<th>Data (48 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data: d9000afa00000000001029000000000000000000000000000000000</td>
</tr>
<tr>
<td>[Length: 48]</td>
</tr>
</tbody>
</table>

- Botnet analysis
- Honeypot setup
- Define input formats for Smart Fuzzing
A Protocol Specification

Message Formats

Message Types | Vocabulary

Behavior Model | Grammar
A Protocol Specification

Reverse Engineering of

- **Vocabulary**: find message type by format similarities
- **Grammar**: generalize to sequence of messages types
A Protocol Specification

Reverse Engineering of

- **Vocabulary**: find message type by *format* similarities
- **Grammar**: generalize to sequence of *messages types*
Format Inference

Determine field boundaries
Format Inference

Determine field boundaries:

Textual protocol (SMTP):
RCPT TO: <twanda@blue6.ex>

Binary protocol (DHCP):
638253633501053604ac140301330400000e10
Format Inference

Determine field boundaries:

**Textual protocol (SMTP):**

```
RCPT TO: <twanda@blue6.ex>
```

**Binary protocol (DHCP):**

```
638253633501053604ac140301330400000e10
```

*Key:*

- **Keyword**
- **Separator**
- **Value**
Format Inference

Determine field boundaries:

Textual protocol (SMTP):

| RCPT | TO: | <twanda@blue6.ex> |

Binary protocol (DHCP):

| 638253633501053604ac140301330400000e10 |

Key:
- Keyword
- Separator
- Value
A New Kind of Feature

One NTP message per row

Byte position

Byte value
A New Kind of Feature

One NTP message per row

Byte position

Byte value

input/ntp_SMIA-20111010_deduped-100.pcap
A New Kind of Feature

One NTP message per row

Byte position

Byte value
A New Kind of Feature

One NTP message per row

Byte position

Byte value
A New Kind of Feature

One NTP message per row
A New Kind of Feature

One NTP message per row

Byte position

Byte value

message

input/ntp_SMIA-20111010_deduped-100.pcap
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240

4 8 12 16 20 24 28 32 36 40 44 48 52 56 60
byte ... 5
Observations

- Intrinsic structure observable

- Messages designed for efficient parsing:
  - fixed-length data types
  - fields not uniformly filled
    - 00 0af8 fe

- Counted numbers have specific variance distribution
Novel Feature to Pinpoint Boundaries

Change in the agreement of bits in consecutive bytes throughout ONE single message!

Deltas of Bit Congruence
Deltas of Bit Congruence

Bit Congruence:

based on similarity measure for bit strings
by Sokal and Michener (1958)
Deltas of Bit Congruence

Bit Congruence:

\[ \Delta_{BC} = \left( BC\left(m^k; m^{k+1}\right); BC\left(m^k; m^{k+1}\right) \right) \]

\[ 0 < k < n \]

Bit Congruence:

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

\[ c_{\text{agree}}(b, \overline{b}): \text{ number of congruent bits for bytes } b \text{ and } \overline{b} \]
Deltas of Bit Congruence

\[ \Delta BC = (BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \overline{b}) \): number of congruent bits for bytes \( b \) and \( \overline{b} \)
Deltas of Bit Congruence

\[ \Delta BC = (BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ BC(b, \bar{b}) = \frac{c_{\text{agree}}(b, \bar{b})}{8} \]

\( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)

\( c_{\text{agree}}(b, \bar{b}) \): number of congruent bits for bytes \( b \) and \( \bar{b} \)
Deltas of Bit Congruence

\[ \Delta \text{BC} = (\text{BC}(m_k, m_{k+1}) - \text{BC}(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ \text{BC}(b, \bar{b}) = \frac{c_{\text{agree}}(b, \bar{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \bar{b}) \): number of congruent bits for bytes \( b \) and \( \bar{b} \)
Deltas of Bit Congruence

\[ \Delta BC = (BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \overline{b}) \): number of congruent bits for bytes \( b \) and \( \overline{b} \)
Deltas of Bit Congruence

\[ \Delta BC = \left( BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k) \right)_{0 < k < n} \]

with

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \overline{b}) \): number of congruent bits for bytes \( b \) and \( \overline{b} \)
Deltas of Bit Congruence

\[ \Delta BC = (BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \overline{b}) \): number of congruent bits for bytes \( b \) and \( \overline{b} \)
Deltas of Bit Congruence

\[ \Delta BC = (BC(m_k, m_{k+1}) - BC(m_{k-1}, m_k))_{0 < k < n} \]

with

\[ BC(b, \overline{b}) = \frac{c_{\text{agree}}(b, \overline{b})}{8} \]

- \( m_k \): Message \( m \)'s byte at position \( k \), \( m \) has length \( n + 1 \)
- \( c_{\text{agree}}(b, \overline{b}) \): number of congruent bits for bytes \( b \) and \( \overline{b} \)
Applying the Feature

Feature $\Delta BC$:
distinctive distribution for binary numbers:

- At field transition: low $\Delta BC$
- Towards field end: high $\Delta BC$
Applying the Feature

Feature $\Delta BC$:
distinctive distribution for binary numbers:

- At field transition: low $\Delta BC$
- Towards field end: high $\Delta BC$
- Gaussian filter $g_\sigma(\cdot)$ to reduce noise
Applying the Feature

Feature $\Delta BC$:
distinctive distribution for binary numbers:

- At field transition: low $\Delta BC$
- Towards field end: high $\Delta BC$
- Gaussian filter $g_\sigma(\cdot)$ to reduce noise

Inflection points of rising edges of $g_\sigma(\Delta BC)$
Value Pattern: Feature of one NTP message
Value Pattern: Feature of one NTP message
Value Pattern: Feature of one NTP message
Value Pattern: Feature of one NTP message
Value Pattern: Feature of one NTP message

![Graph showing periodic and coinciding features with Gaussian-smoothed feature](image-url)
Value Pattern: Feature of one DNS message

![Graph showing byte positions and delta BC values for different DNS messages.](image-url)
Value Pattern: Feature of one DNS message

Except:
- char sequences and high-entropy data
Value Pattern: Feature of one DNS message

Except:
char sequences and high-entropy data

Byte Position
Pinpoint Field Boundaries: DNS

Messages

Aligned Byte Position

4 8 12 92 96
Pinpoint Field Boundaries: NTP
Quantify Format Inference Quality

Validate format inference method:
Measure correctness by benchmarking with a known protocol
Format Match Score

\[
\text{FMS} = \exp \left( - \left( \frac{|R| - |I|}{|R|} \right)^2 \right) \cdot \frac{1}{|R|} \sum_{r \in R} \exp \left( - \left( \frac{\delta_r}{\gamma} \right)^2 \right)
\]

Specificity penalty

Match gain
Format Match Score

$$FMS = \exp \left( - \left( \frac{|R| - |I|}{|R|} \right)^2 \right) \cdot \frac{1}{|R|} \sum_{r \in R} \exp \left( - \left( \frac{\delta_r}{\gamma} \right)^2 \right)$$

Specificity penalty
Match gain

Quality aspects:

$|R|$ Number of real field boundaries

$|I|$ Number of inferred field boundaries
Format Match Score

\[ FMS = \exp \left( - \left( \frac{|R| - |I|}{|R|} \right)^2 \right) \cdot \frac{1}{|R|} \sum_{r \in R} \exp \left( - \left( \frac{\delta_r}{\gamma} \right)^2 \right) \]

\begin{align*}
\text{Specificity penalty} & \quad \text{Match gain} \\
|R| & \quad \text{Number of real field boundaries} \\
|I| & \quad \text{Number of inferred field boundaries} \\
\delta_r & \quad \text{Distance of real boundary } r \text{ from next inferred one} \\
\gamma & \quad \text{Required accuracy}
\end{align*}
Format Match Score

\[ FMS = \exp \left( - \left( \frac{|R| - |I|}{|R|} \right)^2 \right) \cdot \frac{1}{|R|} \sum_{r \in R} \exp \left( - \left( \frac{\delta_r}{\gamma} \right)^2 \right) \]

Specificity penalty

Match gain

Weighted distance
\[ \gamma = 2 \]

\[ \delta_r = \pm \infty \]

exact match

no matching inferred field
Format Match Score

\[ \text{FMS} = \exp \left( - \left( \frac{|R| - |I|}{|R|} \right)^2 \right) \cdot \frac{1}{|R|} \sum_{r \in R} \exp \left( - \left( \frac{\delta_r}{\gamma} \right)^2 \right) \]

Specificity penalty \hspace{1cm} \text{Match gain}

Quantify format correctness
Implementation

NEMESYS

NEtwork MEssage SYntax analysis
NEMESYS Architecture

PCAP file

SpecimenLoader

message

MessageAnalyzer

\( g_\sigma (\Delta BC) + \) inflection point approximation

message format

refinements

MessageSegment
Evaluation Process

inferred MessageSegment

MessageComparator

ParsedMessage

FMS

format comparison

running `tshark` and parsing its output
Evaluation Results: Quality

<table>
<thead>
<tr>
<th></th>
<th>DNS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FMS</strong></td>
<td><strong>100</strong></td>
<td><strong>1000</strong></td>
</tr>
<tr>
<td><strong>best case</strong></td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td><strong>average case</strong></td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Netzob  | NEMESYS
Evaluation Results: Performance

Reduces runtime from exponential to linear
Future Work

Use characteristic features to recognize field data types

- **Integer**: `00 0af8 fe`
- **String**: `69 44 53 00`
- **Padding**: `57 b0 00 00`

- Find more data-type-specific patterns:
  - flags
  - addresses
  - signed numbers
  - floats
  - enumerations

**Message Type Identification:**
Cluster messages on patterns of segment data types
(based on Cui et al., 2007; FieldHunter, 2016)
Conclusion

**NEMESYS:**
Novel method for format inference

- Intrinsic message structure
- Binary protocols
- Abstracting from concrete byte values
- Linear time complexity
Conclusion

**NEMESYS:**
Novel method for format inference

- Intrinsic message structure
- Binary protocols
- Abstracting from concrete byte values
- Linear time complexity

**Format Match Score:**
Quality assessment of format inference methods
THANK YOU!

Questions?

web  uulm.de?kleber
mail  stephan.kleber@uni-ulm.de

Institute of Distributed Systems, Ulm University

web  uulm.de/in/vs
github  github.com/vs-uulm