Backdoor Detection Tools for the Working Analyst

Sam L. Thomas
(based on joint work with Flavio D. Garcia & Tom Chothia)

School of Computer Science
University of Birmingham
Birmingham
United Kingdom
B15 2TT

s.l.thomas@cs.bham.ac.uk

CRYPTACUS Workshop & MC Meeting 2017
An Ideal Situation
An Ideal Situation
An Ideal Situation
An Ideal Situation
A Real-world Situation
A Real-world Situation
A Real-world Situation
Challenge

How do we reduce the manual effort required to identify undocumented functionality and backdoors within software?
Challenge

- How do we reduce the manual effort required to identify undocumented functionality and backdoors within software?
Motivation

- Undocumented functionality? Backdoors?
  - Authentication bypass by “magic” words.
  - Hard-coded credential checks.
  - Additional protocol messages that activate unexpected functionality.
  - Common services that perform non-standard functionality.
Focus on IoT devices:

- Lots of devices, lots of firmware, different architectures.
- Devices are attached to our networks often without regard for how secure they are.
- Can’t manually analyse every firmware image.
Tools

- **HumIDIFy**: detects undocumented, non-standard functionality in common services.
- **Stringer**: detects hard-coded credentials and undocumented protocol messages.
Objective

Both tools:

- Lightweight analysis.
- Reduce time required and expertise to perform analysis.
HumIDIFy
Method – Overview

- Uses machine learning to identify common executable classes (e.g. FTP server, Web server, ...).
- Tests to see if these identified common services perform more than their expected functionality (e.g. a Web server that listens for commands on a high UDP port and executes them as root on the device).
Method – Machine Learning

- Uses semi-supervised learning: train a classifier using some labelled instances and a larger amount of unlabelled instances.
- Uses an algorithm called self-training: iterates until some stability is reached on the performance of the classifier.
- On real-world test data (manually labelled, independent from training set): 96.4523% correctness.
Method – Testing Functionality

- High-level domain-specific language (DSL) to encode expected program functionality.
- DSL interpreter processes *functionality profile* and target executable.
- Have a *functionality profile* for each type of common service – they have known, well-defined behaviour.
Example rules written in the DSL:

```plaintext
rule handles_socket() =
    function_ref("socket")

rule handles_tcp() =
    handles_socket() && (function_ref("recv") ||
    function_ref("send"))
```
Tenda Router web-server analysis with HumIDIFy:

$ ./HumIDIFy model/BayesNet httpd

] HumIDIFy: version 1.0 ,-
]-------------------------|-
[i] performing feature extraction...
[i] classifying binary...
-> File : httpd
-> Profile : webserver (with confidence 100.00%)
[i] checking binary’s functionality...
-> Warning : udp-based api usage detected
-> Judgement : potentially anomalous
Method – Overview

- Assigns scores to static data and functions to indicate their relevance/potential.
- Generates a summary report of the executable using scoring for faster, simpler analysis.
Method – Overview

- Automatically identifies potential static data comparison functions.
- Extracts the arguments passed to those functions when the function call influences a branch condition.
- Maximises scores to static data based on how much CFG functionality they guard.
$ ./Stringer td3250

*** attempting to locate comparison functions...

[h] 15669 functions analysed; comparison functions:
[c] strcmp (1388.100000)
[c] strncmp (773.326250)
...

*** computing scores...
...

[f] 556.59: _ZN9CLoginDlg5LogInEPKcS1_b
  288.35: admin (via: strcmp)
  60.92: ppttzz51shezhi (via: strcmp)
  49.83: 6036logo (via: strcmp)
...

Case studies
Identification of hard-coded credential pair in Ray Sharp DVR firmware:

<table>
<thead>
<tr>
<th>Comparison Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcmp</td>
<td>5170.30</td>
</tr>
<tr>
<td>sub_1C7EC (strcmp wrapper)</td>
<td>1351.96</td>
</tr>
<tr>
<td>strncmp</td>
<td>1109.73</td>
</tr>
<tr>
<td>strstr</td>
<td>353.93</td>
</tr>
<tr>
<td>memcmp</td>
<td>222.00</td>
</tr>
</tbody>
</table>

Label | Score | Static Data | Function | Depends |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.23</td>
<td>664225</td>
<td>strcmp</td>
<td>{}</td>
</tr>
<tr>
<td>2</td>
<td>2.77</td>
<td>root</td>
<td>strcmp</td>
<td>{664225}</td>
</tr>
</tbody>
</table>

Diagram illustrating the comparison functions and their scores.
Identification of a hard-coded credential backdoor in DVR firmware – different behaviour for each hardcoded password:

<table>
<thead>
<tr>
<th>Comparison Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strcmp</code></td>
<td>1464.70</td>
</tr>
<tr>
<td><code>strncmp</code></td>
<td>779.33</td>
</tr>
<tr>
<td><code>CRYPTO_malloc</code> (FP)</td>
<td>685.10</td>
</tr>
<tr>
<td><code>_ZNKSSs7compareEPKc</code></td>
<td>376.20</td>
</tr>
<tr>
<td><code>strstr</code></td>
<td>306.00</td>
</tr>
<tr>
<td><code>strcasecmp</code></td>
<td>196.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Score</th>
<th>Static Data</th>
<th>Function</th>
<th>Depends</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>171.39</td>
<td><code>admin</code></td>
<td><code>strcmp</code></td>
<td><code>{[]}</code></td>
</tr>
<tr>
<td>2</td>
<td>58.92</td>
<td><code>ppttzz51shezhi</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
<tr>
<td>3</td>
<td>45.13</td>
<td><code>6036logo</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
<tr>
<td>4</td>
<td>42.14</td>
<td><code>6036adws</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
<tr>
<td>5</td>
<td>37.54</td>
<td><code>6036huanyuan</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
<tr>
<td>6</td>
<td>35.21</td>
<td><code>6036market</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
<tr>
<td>7</td>
<td>31.05</td>
<td><code>jiamijiami6036</code></td>
<td><code>strcmp</code></td>
<td><code>{[admin]}</code></td>
</tr>
</tbody>
</table>
Web-server with thread running UDP-based service executing user-input commands, unauthenticated as root user:

```bash
./HumIDIFy model/BayesNet _US_W302RRA_.../bin/httpd
]] HumIDIFy: version 1.0 ,-.
]------------------------|--'
[i] performing feature extraction...
[i] classifying binary...

-> File : _US_W302RRA_.../bin/httpd
-> Profile : webserver (with confidence 100.00%)

[i] checking binary’s functionality...

-> Warning : udp-based api usage detected

-> Judgement : potentially anomalous
Tenda Web-server “Management Service” (cont.)

Web-server with thread running UDP-based service executing user-input commands, unauthenticated as root user:
TrendNet HTTP Authentication with Hard-coded Credentials

HTTP authentication check with comparison against hard-coded credential values:

<table>
<thead>
<tr>
<th>Comparison Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcmp</td>
<td>1635.01</td>
</tr>
<tr>
<td>strstr</td>
<td>481.20</td>
</tr>
<tr>
<td>nvram_get (FP)</td>
<td>413.10</td>
</tr>
<tr>
<td>strncmp</td>
<td>265.45</td>
</tr>
<tr>
<td>sub_A2D0 (FP)</td>
<td>131.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Static Data</th>
<th>Score</th>
<th>Function</th>
<th>Depends</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptyuserrrrrrrrrrrrr</td>
<td>132.17</td>
<td>strcmp</td>
<td>{...}</td>
</tr>
<tr>
<td>emptypasswordddddddd</td>
<td>128.61</td>
<td>strcmp</td>
<td>{..., emptyuserrrrrrrrrrrrrrrr}</td>
</tr>
</tbody>
</table>
We are also able to recover the command sets of proprietary protocols, in this case a SOAP command set:

<table>
<thead>
<tr>
<th>Comparison Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcmp</td>
<td>380.52</td>
</tr>
<tr>
<td>safestrcmp (custom string comparison)</td>
<td>221.00</td>
</tr>
<tr>
<td>strstr</td>
<td>185.00</td>
</tr>
<tr>
<td>strcasecmp</td>
<td>184.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Score</th>
<th>Static Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.64</td>
<td>EnableTrafficMeter</td>
</tr>
<tr>
<td>2</td>
<td>7.64</td>
<td>SetTrafficMeterOptions</td>
</tr>
<tr>
<td>3</td>
<td>7.64</td>
<td>SetGuestAccessEnabled</td>
</tr>
<tr>
<td>4</td>
<td>7.64</td>
<td>SetGuestAccessEnabled2</td>
</tr>
<tr>
<td>5</td>
<td>7.64</td>
<td>SetGuestAccessNetwork</td>
</tr>
<tr>
<td>6</td>
<td>7.64</td>
<td>SetWLANNoSecurity</td>
</tr>
<tr>
<td>7</td>
<td>7.64</td>
<td>SetWLANWPAPSKByPassphrase</td>
</tr>
</tbody>
</table>
Performance
HumIDIFy

- Attribute extraction: 1.31s.
- Classification of single binary: 0.291s (not including time taken to invoke the Java virtual machine).
- Performance of DSL interpreter is dependent upon the complexity of the binary under analysis (number of functions and complexity of those functions): 1.53s on average.
- Time to process an “average” firmware image: 970.61s.
- Performance analysis does not take into account the human factor in final manual analysis.
• Average processing time for a binary: 1.3s.
• Some take longer - depends upon number of functions and CFG complexity:
  Q-See DVR firmware took 46.043 with 15,669 functions.
Conclusion

- Runtime of both tools satisfies lightweight property: each tool takes seconds to perform analysis.
- Successfully identified a number of backdoors and instances of undocumented functionality.