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

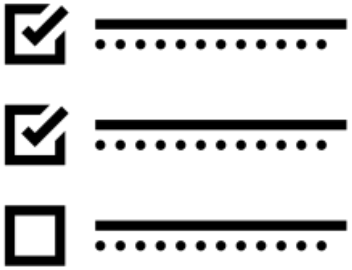
s.l.thomas@cs.bham.ac.uk

CRYPTACUS Workshop & MC Meeting 2017

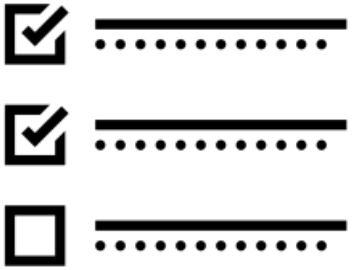
An Ideal Situation

An Ideal Situation

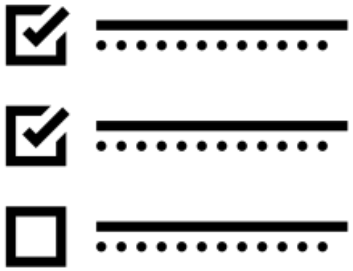
An Ideal Situation



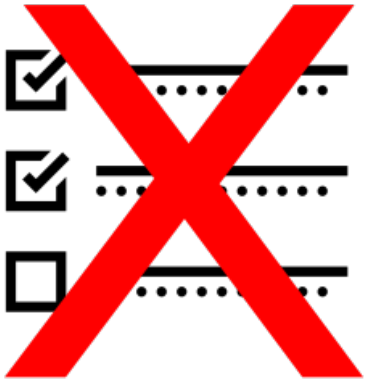
An Ideal Situation



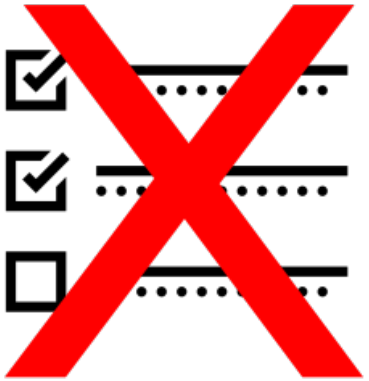
A Real-world Situation



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.

Application

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.

Method – Testing Functionality (cont.)

Example rules written in the DSL:

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

rule handles_tcp() =
    handles_socket() && (function_ref("recv") ||
                         function_ref("send"))
```

Usage

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
```

Stringer

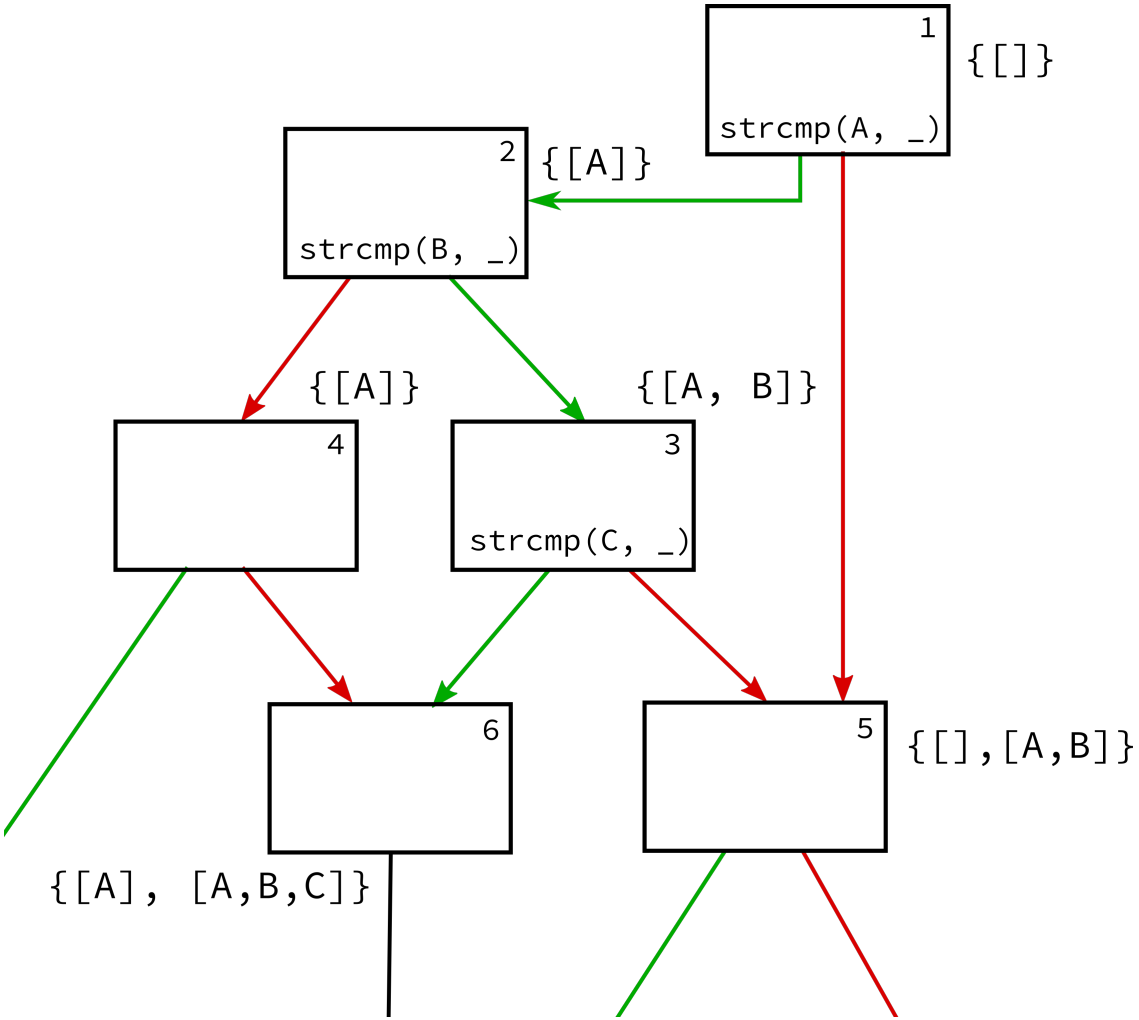
Method – Overview

- 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*.

Method – Score Assignment



Usage

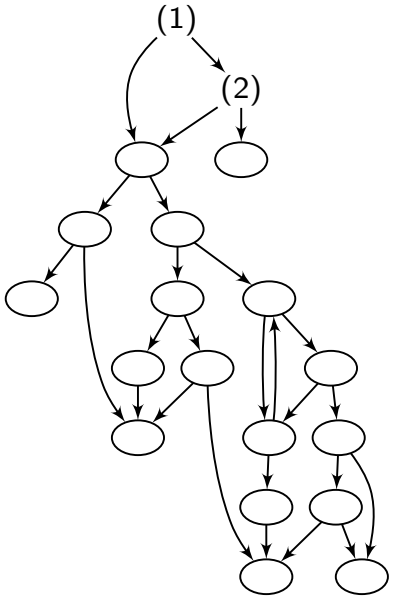
```
$ ./Stringer td3250
*** attempting to locate comparison functions...
[h] 15669 functions analysed; comparison functions:
[c] strcmp (1388.100000)
[c] strncmp (773.326250)
...
*** computing scores...
...
[f] 556.59: _ZN9CLoginDlg5LogInEPKcS1_b
  288.35: admin (via: strcmp)
  60.92: ppttzz51shezhi (via: strcmp)
  49.83: 6036logo (via: strcmp)
  ...
```

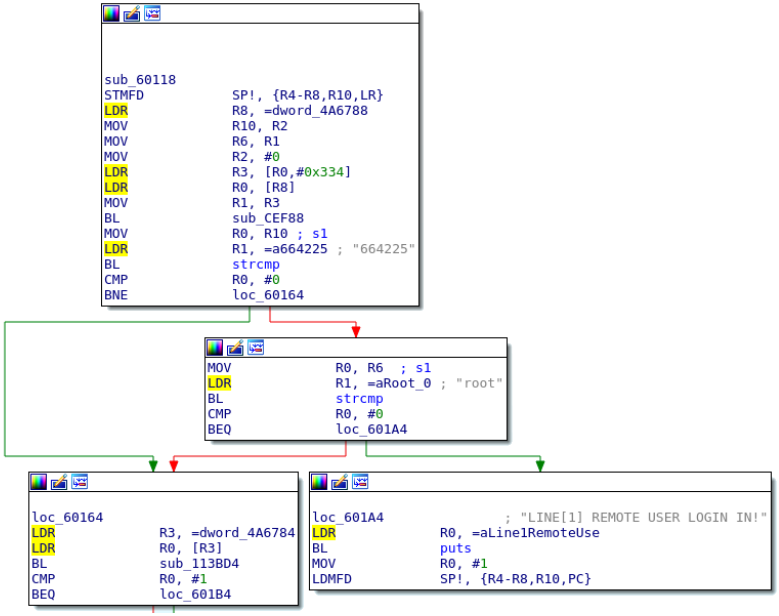
Case studies

Hard-coded Credentials in Ray Sharp DVR Firmware

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

Comparison Function	Score
`strcmp`	5170.30
`sub_1C7EC` (`strcmp` wrapper)	1351.96
`strncmp`	1109.73
`strstr`	353.93
`memcmp`	222.00





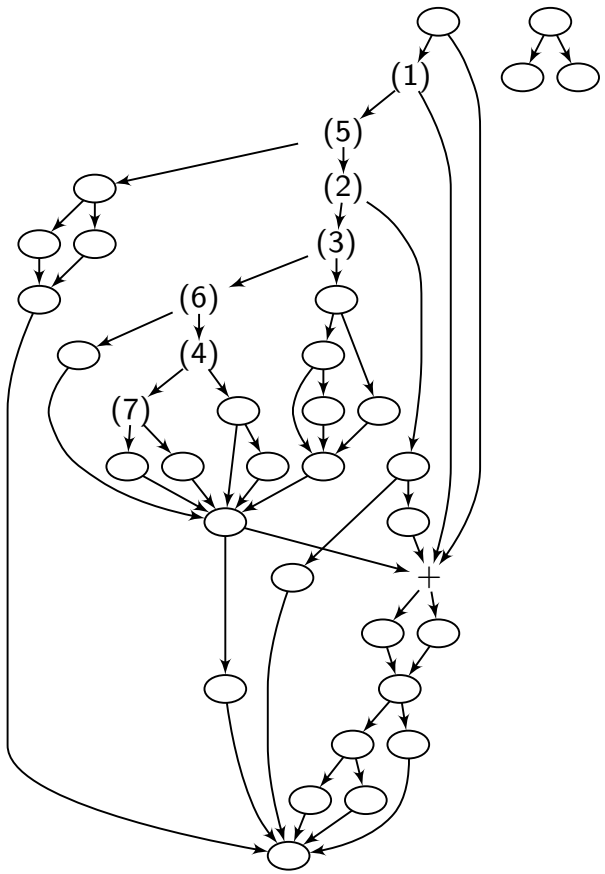
Label	Score	Static Data	Function	Depends
1	30.23	664225	`strcmp`	{[]}
2	2.77	`root`	`strcmp`	{[664225]}

Hard-coded Credentials in Q-See DVR Firmware

Identification of a hard-coded credential backdoor in DVR firmware – different behaviour for each hardcoded password:

Comparison Function	Score
strcmp	1464.70
strncmp	779.33
CRYPTO_malloc (FP)	685.10
_ZNKSs7compareEPKc	376.20
strstr	306.00
strcasecmp	196.00

Label	Score	Static Data	Function	Depends
1	171.39	admin	strcmp	{[]}
2	58.92	ppttzz51shezhi	strcmp	{[admin]}
3	45.13	6036logo	strcmp	{[admin]}
4	42.14	6036adws	strcmp	{[admin]}
5	37.54	6036huanyuan	strcmp	{[admin]}
6	35.21	6036market	strcmp	{[admin]}
7	31.05	jiamijiami6036	strcmp	{[admin]}



Tenda Web-server “Management Service”

Web-server with thread running UDP-based service executing user-input commands, unauthenticated as root user:

```
./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:

```
nop
lw      $gp, 0xA98+var_A80($sp)
move    $a0, $s0
la      $t9, inet_addr
nop
jalr    $t9 ; inet_addr
nop
lw      $gp, 0xA98+var_A80($sp)
li      $a0, 2
li      $a1, 1
move    $a2, $zero
sw      $v0, 0xA98+var_A74($sp)
la      $t9, socket
nop
jalr    $t9 ; socket
nop
lw      $gp, 0xA98+var_A80($sp)
nop
la      $a0, aRenable    # "rEnable"
nop
addiu   $a0, (aMfgthreadSocke - 0x480000)  # "MfgThread socket error."
bltz    $v0, loc_433C9C
move    $s1, $v0
```

```
move    $a0, $v0
addiu   $a1, $sp, 0xA98+var_A78
li      $a2, 0x10
la      $t9, bind
nop
jalr    $t9 ; bind
nop
lw      $gp, 0xA98+var_A80($sp)
bltz    $v0, loc_433C90
li      $s3, 0x10
```

```
li      $a2, 0x80
move    $a3, $zero
sw      $s3, 0xA98+var_40($sp)
sw      $s4, 0xA98+var_A88($sp)
sw      $s6, 0xA98+var_A84($sp)
la      $t9, recvfrom
nop
jalr    $t9 ; recvfrom
nop
lw      $gp, 0xA98+var_A80($sp)
blez    $v0, loc_433C70
```

```
loc_433B34:
move    $a0, $s2
move    $a1, $s7
li      $a2, 0x800
la      $t9, call_shell
nop
jalr    $t9 ; call_shell
nop
lw      $gp, 0xA98+var_A80($sp)
blez    $v0, loc_433958
move    $s0, $v0
```

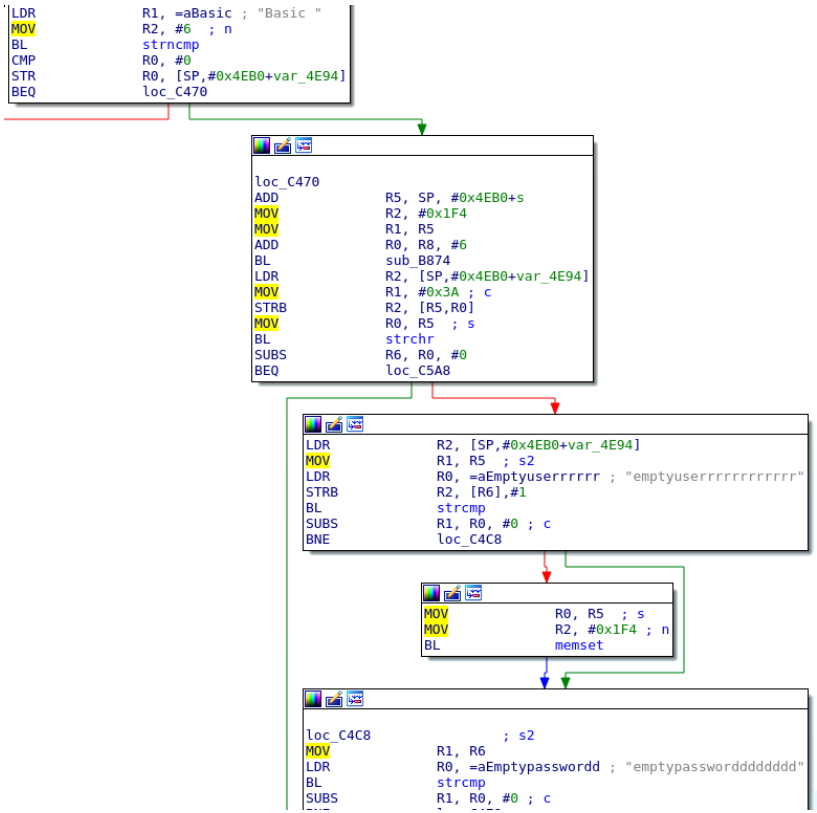
```
.globl call_shell
call_shell:

var_128= -0x128
var_120= -0x120
var_20= -0x20
var_1C= -0x1C
var_18= -0x18
var_14= -0x14
var_10= -0x10
var_C= -0xC
var_8= -8
var_4= -4

li      $gp, 0x8B93C
addu    $gp, $t9
addiu   $sp, -0x138
sw      $gp, 0x138+var_128($sp)
sw      $s5, 0x138+var_C($sp)
move    $s5, $a1
la      $a1, aRenable    # "rEnable"
nop
addiu   $a1, (aR_0 - 0x480000)  # "r"
sw      $s4, 0x138+var_10($sp)
sw      $s3, 0x138+var_14($sp)
sw      $s2, 0x138+var_18($sp)
sw      $ra, 0x138+var_4($sp)
sw      $gp, 0x138+var_8($sp)
sw      $s1, 0x138+var_1C($sp)
sw      $s0, 0x138+var_20($sp)
move    $s4, $a2
la      $t9, popen
nop
jalr    $t9 ; popen
nop
lw      $gp, 0x138+var_128($sp)
move    $s3, $v0
move    $s2, $zero
beqz    $s3, loc_43EFBC
li      $v0, 0xFFFFFFFF
```

TrendNet HTTP Authentication with Hard-coded Credentials

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



Comparison Function	Score
`strcmp`	1635.01
`strstr`	481.20
`nvram_get` (FP)	413.10
`strncmp`	265.45
`sub_A2D0` (FP)	131.00

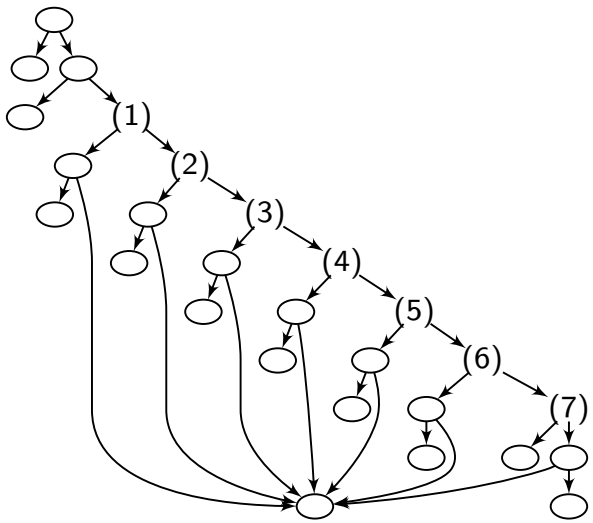
Static Data	Score	Function	Depends
`emptyuserrrrrrrrrrrr`	132.17	`strcmp`	$\{\ldots\}$
`emptypassworddddddddd`	128.61	`strcmp`	$\{[\ldots, \texttt{emptyuserrrrrrrrrrrr}]\}$

Recovery of SOAP-based Command Set

We are also able to recover the command sets of proprietary protocols, in this case a SOAP command set:

Comparison Function	Score
`strcmp`	380.52
`safestrcmp` (custom string comparison)	221.00
`strstr`	185.00
`strcasecmp`	184.00

Label	Score	Static Data
1	7.64	`EnableTrafficMeter`
2	7.64	`SetTrafficMeterOptions`
3	7.64	`SetGuestAccessEnabled`
4	7.64	`SetGuestAccessEnabled2`
5	7.64	`SetGuestAccessNetwork`
6	7.64	`SetWLANNoSecurity`
7	7.64	`SetWLANWPAPSKByPassphrase`



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.

Stringer

- 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.
- Sucessfully identified a number of backdoors and instances of undocumented functionality.

Q & A