HumIDIFy: A Tool for Hidden Functionality Detection in Firmware

Sam L. Thomas, Flavio D. Garcia, Tom Chothia

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

{s.l.thomas,f.garcia,t.p.chothia}@cs.bham.ac.uk

14th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2017)

July 7th 2017

UNIVERSITY OF BIRMINGHAM

Overview

- COTS embedded device security is a disaster:
  - Poor coding practices.
  - Internet-facing “debug” interfaces.
  - Hard-coded credential checks.
  - Additional, hidden services.

UNIVERSITY OF BIRMINGHAM

Overview (cont.)

- Lots of devices, lots of firmware (mostly Linux based).
- Multiple architectures (ARM, MIPS, PPC, etc.).
- Multiple firmware versions for each device.
- Manual analysis takes significant time and expertise.

UNIVERSITY OF BIRMINGHAM

Existing Approaches

- A Large Scale Analysis of the Security of Embedded Firmwares (Costin, et al.).
- Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces (Costin, et al.).
- Firmalice - Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware (Shoshitaishvili, et al.).

UNIVERSITY OF BIRMINGHAM

HumIDIFy

UNIVERSITY OF BIRMINGHAM

Research Objective

- Address the problem of detecting additional, unexpected functionality in common services.
- A lightweight way to:
  - Identify classes of program functionality.
  - Identify anomalous functionality within those classes.

UNIVERSITY OF BIRMINGHAM

Research Method

- A means to identify classes of program functionality:

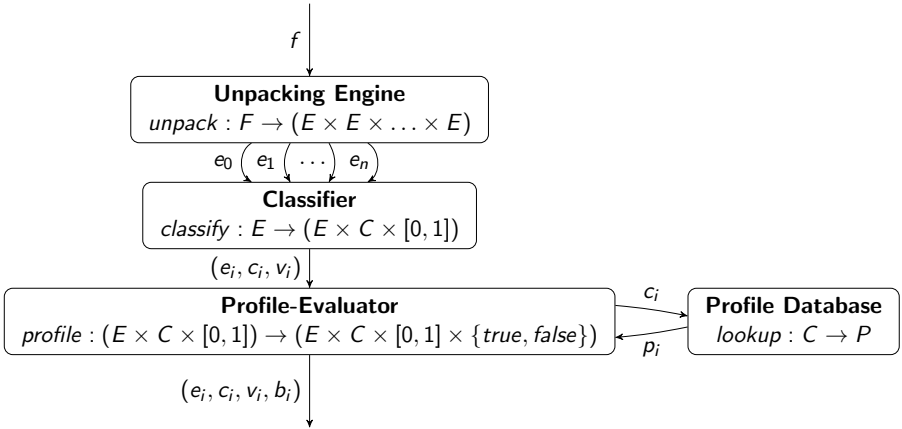
  Use machine learning to identify classes of programs.

- A means to identify anomalous functionality:

  Use a DSL to define *expected* program functionality.

UNIVERSITY OF BIRMINGHAM

HumIDIFy



UNIVERSITY OF BIRMINGHAM

Data-set Composition

- Collected set of firmware (15,438 images) from multiple vendors (30).
- Unpacked firmware reduced the data set to 7590 images (2,451,532 binaries).
- 3-way split of data set into training and validation sets and a further set for evaluation.

UNIVERSITY OF BIRMINGHAM

Classifying Binary Functionality

UNIVERSITY OF BIRMINGHAM

Attribute Selection

- Focus on features that are consistent across different architectures:
  - Strings.
  - Imported API names.
- Labels describe broad functionality categories (e.g. web server, etc.).
- Use `CfsSubsetEval` with `BestFirst` ranking.
- E.g. $\langle 1, 1, 0, \ldots, \texttt{web-server} \rangle$.

UNIVERSITY OF BIRMINGHAM

Classifier Construction

- Semi-supervised learning:
  - What?
  - Why?

UNIVERSITY OF BIRMINGHAM

Supervised Algorithm Evaluation

We first evaluated a number of supervised algorithms and selected the most optimal in terms of classification rate and time:

Classifier	Correct (%)	Time (s)
BayesNet	**88.4848**	**0.00**
NaiveBayes	79.3939	0.01
IBk	84.2424	0.00
KStar	84.2424	0.00
LWL	51.5152	0.00
JRip	66.6667	0.08
OneR	21.2121	0.00
PART	77.5758	0.04

Classifier	Correct (%)	Time (s)
ZeroR	10.9091	0.00
DecisionStump	20.6061	0.00
HoeffdingTree	79.3939	0.00
J48	76.9697	0.00
LMT	85.4545	0.90
RandomForest	88.4848	0.11
RandomTree	78.7879	0.00
REPTree	64.8485	0.03

UNIVERSITY OF BIRMINGHAM

Bounded Self-Training

function BOUNDEDSELFTRAINING(labelledData, unlabelledData, v, bound)
  L ← labelledData, U ← unlabelledData, k ← 0
  loop
    train f from L using supervised learning
    (k', L', U') ← apply f to unlabelled instances in U where u ∈ U' if CONFIDENCE(f(u)) ≥ v
    if U = U' ∨ k' − k ≤ bound **then return** f
    end if
    k ← k', L ← L', U ← U'
  end loop
end function

Iteration	1	2	3	4	5	6	7	8
Correct (%)	88.4848	95.4819	97.0760	97.9021	98.5462	99.2366	99.3256	**99.3691**

Binary Functionality Description Language

UNIVERSITY OF BIRMINGHAM

Binary Functionality Description Language

A simple domain specific language to express properties of programs:

rule uses_udp() = **exists** socket(*domain:int*, *type:int*, *protocol:int*) ⇒
if architecture("MIPS") **then** *type* == 2 **else** *type* == 1

rule may_read_files() = **exists** fopen(*filename:string*, *mode:string*) ⇒
(*mode* == "r" || *mode* == "r+" || *mode* == "w+" || *mode* == "a+")

UNIVERSITY OF BIRMINGHAM

Binary Functionality Description Language

- BFDL is used to define the expected properties of a functionality class from the classifier.
- It is based upon a simple grammar that allows user-defined rules matching over both functional properties of code and meta data found within the binary being analysed.

UNIVERSITY OF BIRMINGHAM

Binary Functionality Description Language

Features built-in rules to identify:

- Strings (and if they are referenced within code):
  `string_exists`, `string_ref`
- Function imports and exports (and if they are referenced within code):
  `import_exists`, `export_exists`, `function_ref`
- Function usage - if and how a function is used (and analyse the properties of the arguments passed to it):
  `exists f(x:int, ...)` $\Rightarrow$ `(x = ...)`, `forall f(...)` $\Rightarrow$ `...`

UNIVERSITY OF BIRMINGHAM

Binary Functionality Description Language

Examples

import "prelude.bfdl"

rule web_server() = uses_tcp() && !uses_udp() && may_read_write_files()
&& !outgoing_socket()

rule puts_x(*x:string*) = **exists** puts(*v:string*) ⇒ *v* == *x*

UNIVERSITY OF BIRMINGHAM

Expected Functionality Classifier + Functionality Checker $\approx$ Anomalous Functionality Detector

UNIVERSITY OF BIRMINGHAM

Evaluation & Results

UNIVERSITY OF BIRMINGHAM

Evaluation of Classifier

For a total of 24 different functionality classes:

- Classification rate of 99.3692% on training data
- Classification rate of 96.4523% on separate test data (manually labelled - 451 binaries)

For the most common services our classifier is highly effective in assigning the correct functional class to a given binary.

UNIVERSITY OF BIRMINGHAM

Evaluation of Classifier

Misclassified instances were generally due to overlapping functionality:

- `busybox` implements a large amount of diverse functionality.
- API usage overlapping in network-based services caused some mislabelling.
- Most commonly mislabelled functionality class "nvram-get-set" is a label describing binaries that perform reads/writes from the NVRAM, usually used to preserve user configuration data. This was largely due to how device vendors implement such functionality: some use calls to external programs (e.g. `nvram-get`, `nvram-set`), others implement the functionality directly.

UNIVERSITY OF BIRMINGHAM

Evaluation of HumIDIFy

- Evaluated on "artificial instances": we added an extremely simple UDP-based backdoor in `mini_httpd` and `utelnetd`.
- We tested unmodified instances of each using HumIDIFy to observe the classification (both classified correctly).
- Performed the same classification attempt upon the modified binaries: produce the same classification and feature vectors.
- Both unmodified binaries are passed as fine, both modified binaries are detected as anomalous – both fail to meet their expected functionality profile.

UNIVERSITY OF BIRMINGHAM

Evaluation of HumIDIFy on Real World Firmware

From 392 unique binaries from 100 firmware images, 9 were flagged as anomalous by HumIDIFy:

- A web server containing a previously documented "management interface" backdoor providing shell execution upon the device.
- A web server with a built-in DNS resolver.
- Custom service implementing an Internet telephony proxy detected as a TCP daemon, but supported UDP as a means of data exchange.
- A custom service implementing HTTP proxy functionality, part of Trent Micro kernel engine additionally using UDP to communicate.

UNIVERSITY OF BIRMINGHAM

Run-time Performance

Average run-time performance statistics:

- Attribute extraction: 1.31s.
- Classification of single binary: 0.291s (not including time taken to invoke the Java virtual machine).
- Performance of DSL evaluator is dependent upon the complexity of the binary under analysis (number of functions and complexity of the 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.

UNIVERSITY OF BIRMINGHAM

Limitations & Security Analysis

Can we evade HumIDIFy?

- Current analysis relies on ability to extract imported symbols.
- We look for a specific class of unexpected functionality.

UNIVERSITY OF BIRMINGHAM

Limitations & Security Analysis (cont.)

- What about binaries that deliberately attempt to masquerade as something else?

UNIVERSITY OF BIRMINGHAM

Conclusion

- We construct a classifier to identify functionality in common services in Linux-based firmware.
- We develop a domain specific language to define the expected functionality of such common services.
- By combining the two components we are able to identify common services that exhibit anomalous functionality.

UNIVERSITY OF BIRMINGHAM

Questions?

UNIVERSITY OF BIRMINGHAM