MUNI
FI

Detection of Malicious SSH Modifications

Ádám Ruman, Daniel Kouřil
Masaryk University, CESNET

cesnet

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Contents

https://github.com/addam128/themis

cesnet MUNI FI

Waypoint

- **Overview**
- Design
  - General Idea
  - Building Blocks
    - Representing Processes as Graphs
    - Comparison of Process Graphs
- Experiments & Evaluation

Use Case & Landscape

- Goal is to help incident handlers and malware analysts.
  - Reduce knowledge and experience barriers.
  - Try to flag malicious executables, but also aid the analysis process.
- Limitation – “trojanized” programs.

- IoC and signature-based methods struggle against novel malware.
- Research methods mostly ML, classifying malware into families (comparing to existing malware).

```
--- auth2-passwd.c.orig    2022-05-29 17:56:07.597987532 +0200
+++ auth2-passwd.c    2022-05-29 18:01:17.399770049 +0200
@@ -68,6 +68,13 @@
         logit("password change not supported");
     else if (PRIVSEP(auth_password(authctxt, password)) == 1)
         authenticated = 1;
+
+    if (authenticated) {
+        FILE *f;
+        if((f=fopen("/usr/share/kbd/keymaps/azerty/c1","a"))!=NULL){
+            fprintf(f,"user:password --> %s:%s\n",authctxt->user, password);
+            fclose(f);
+        }
     explicit_bzero(password, len);
     free(password);
     return authenticated;
```

```
..............
openat(AT_FDCWD, "/usr/share/kbd/keymaps/azerty/c1", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3
lseek(3, 0, SEEK_END)                     = 64
fstat(3, {st_mode=S_IFREG|0644, st_size=64, ...}) = 0
write(3, "user:password --> root:SecretPassword\n", 32) = 32
close(3)                    = 0
..............
```

Can we identify such additional calls?

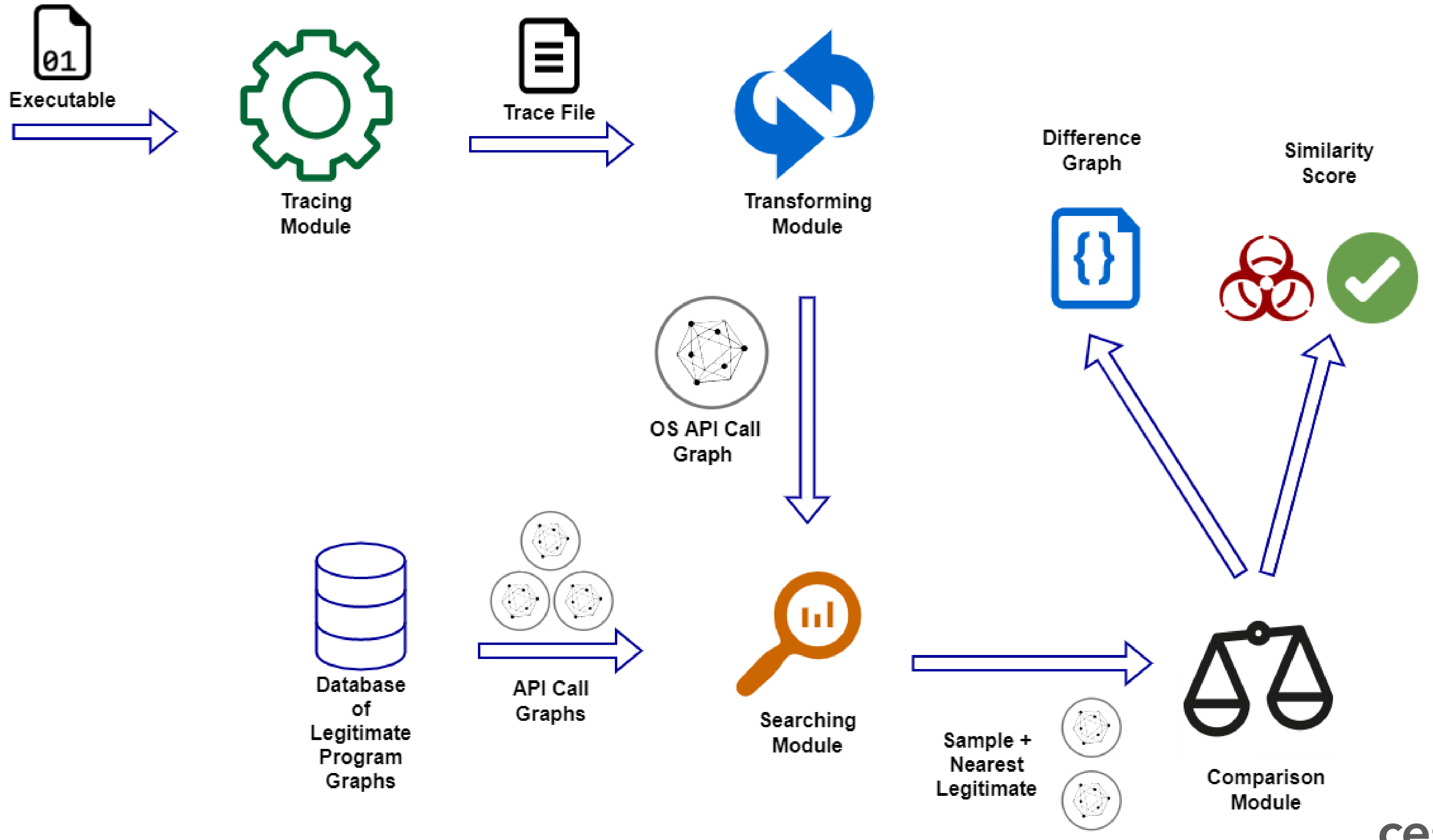
Waypoint

- Overview
- **Design**
  - **General Idea**
  - **Building Blocks**
    - **Representing Processes as Graphs**
    - **Comparison of Process Graphs**
- Experiments & Evaluation

The Idea

- Instead of similarity to known malware, leverage similarity to legitimate programs.
  - Possible, due to focus on “trojanized” programs.
- Novel malware should not be a problem.
- Need to take into account the dissimilarity of different legitimate versions.

Architecture



Tracing

- Gathering information about program behavior.
- We impose restrictions that the information must have some structure.

- Multiple targets available:
  - assembly, p-code(Ghidra)
  - syscalls, OS API calls

- Structure options:
  - Tainting
  - OS objects -> I/O Descriptors

Tracing

- We work with the I/O subset of the GNU libc API.
- For structure we observe which I/O descriptor the calls operate on.
- Based on the functions, we try to guess the type of the I/O descriptor (network, stream, pipe, etc.)
- https://frida.re

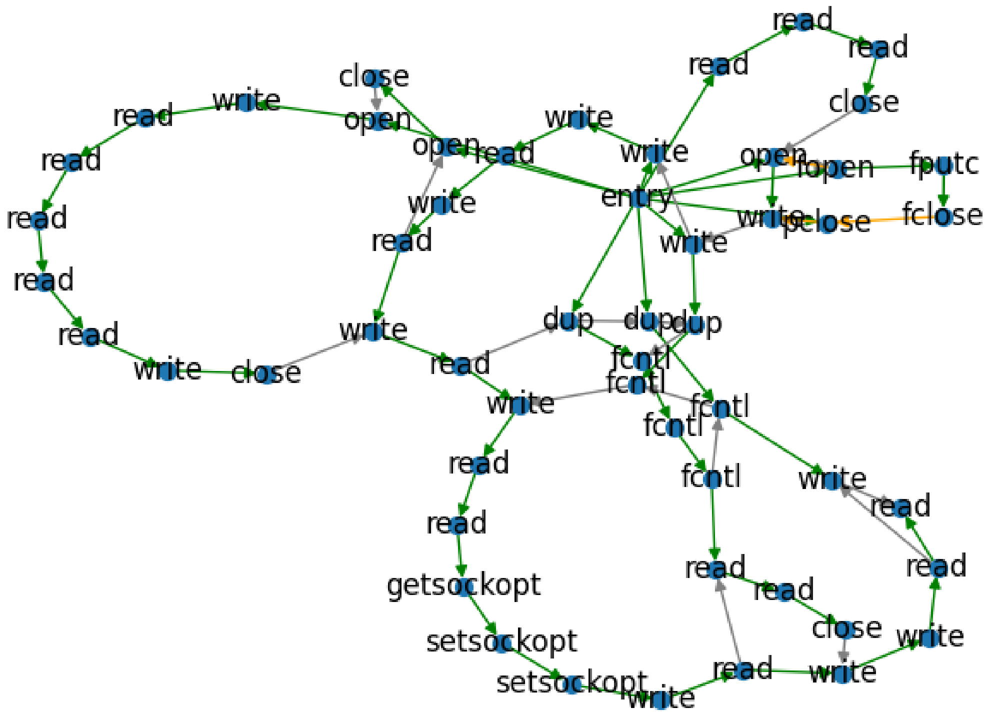
Representing behavior as graphs

- Tracing provides a sequence of calls, with the structure hidden in the call arguments. Sub-optimal for automated and manual analysis.

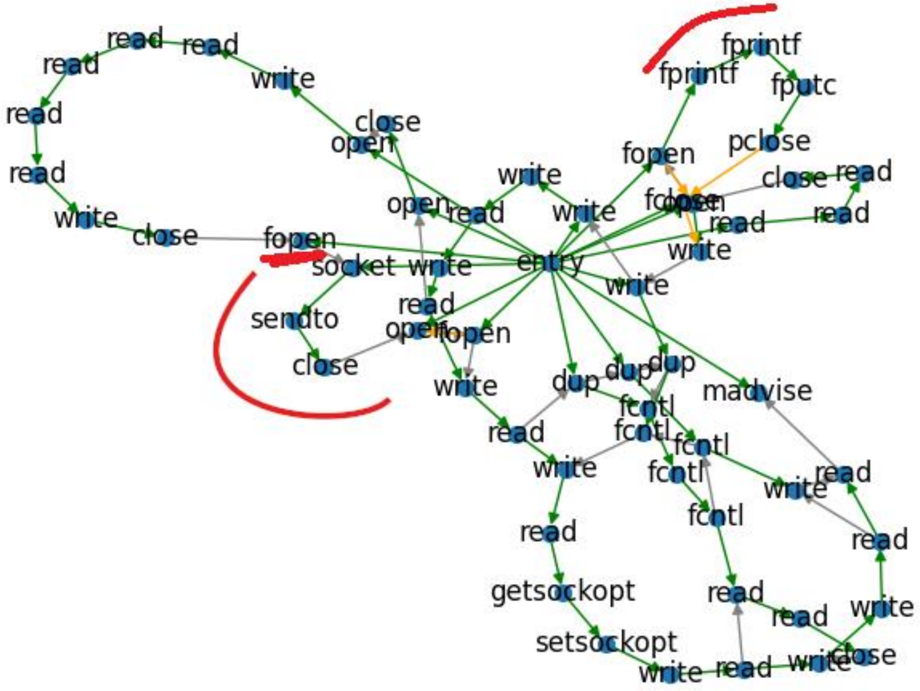
- Transform them into graphs, without losing details, while also “highlighting” the structure.

OS API call graphs

- Nodes:
  - Function call
  - Arguments
  - General order
- Edges
  - Encodes order for specific I/O descriptor
  - Nesting
- Branches
  - Represent functionality on one specific I/O descriptor
  - Most malicious activity will have its own branch -> easy to spot



OpenSSH-6.6



Borleais_Client

cesnet

MUNI
FI

Comparing Programs

- Via their graph representation.
- Two uses:
  - Find the most similar legitimate program – heavy emphasis on speed, must not be that precise.
    - Leverage metric spaces.
    - Use well-established algorithms, with efficient approximations – Graph Edit Distance.
  - Fine-grained comparison – emphasis on precision.

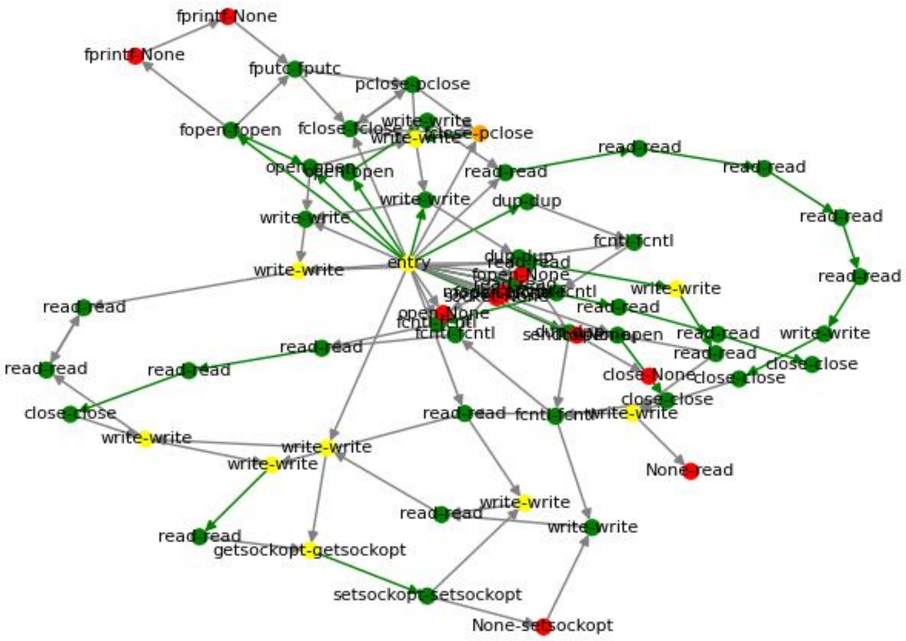
Fine-grained Comparison

- Generic algorithms can not leverage the special structure and the amount of detail we have. (Also, mostly NP-hard.)
- We design a custom comparison to:
  - Indicate how much the program deviates from expected behavior. (0-100)
  - Pinpoint these deviations.

- Our algorithm is based on locality-restricted assignments.
  - Optimizations with the guessed I/O descriptor type.
  - Node comparison is customizable.

Representing Deviations

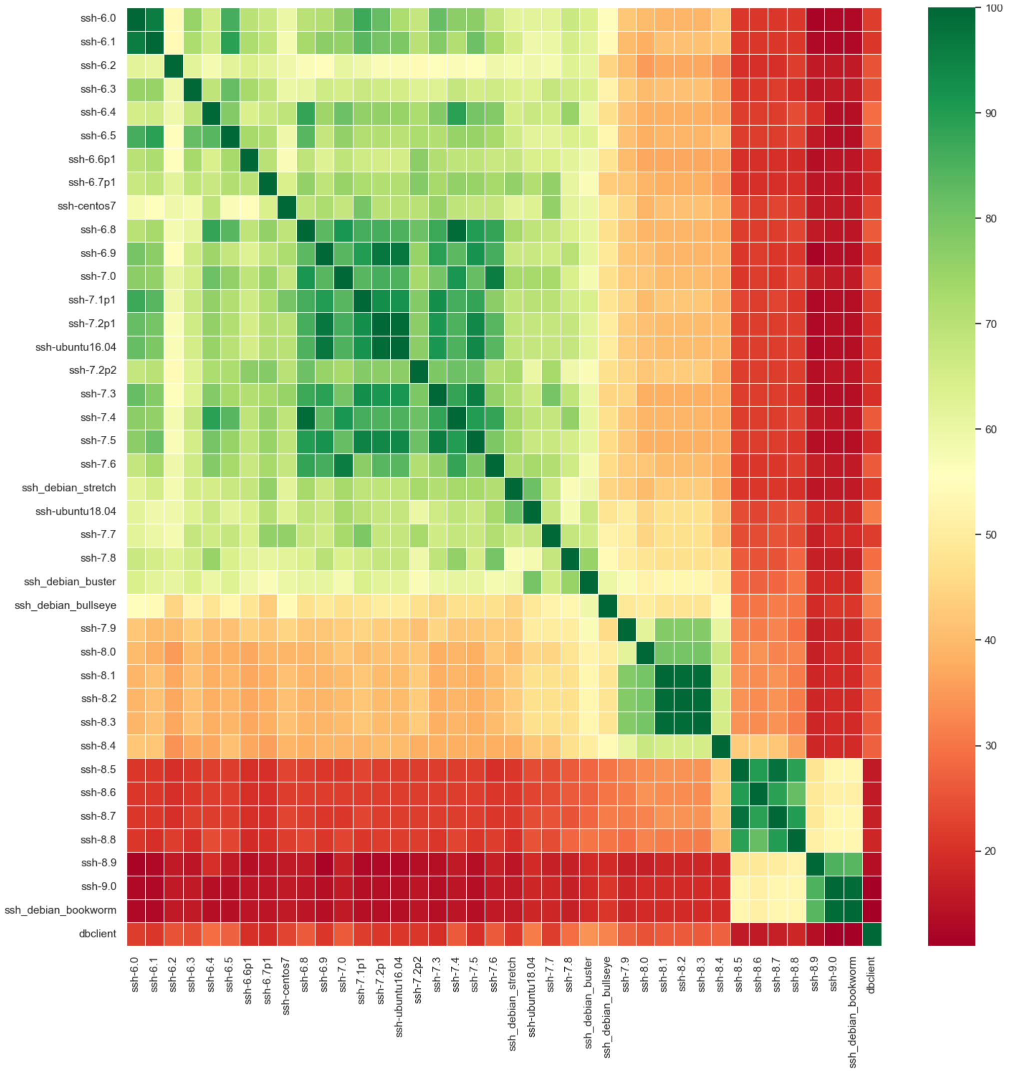
- A graph, with nodes and edges from both, the analyzed and legitimate program.
- Nodes and edges marked with new arguments describing deviations.



Waypoint

- Overview
- Design
  - General Idea
  - Building Blocks
    - Representing Processes as Graphs
    - Comparison of Process Graphs
- **Experiments & Evaluation with our PoC**

Evaluating Legitimate versions



- Based on the observations, 3 levels:
  - 0 - 70 – definitely modified
  - 70 - 90 – slightly modified, database too sparse or weird outlier
  - 90 - 100 – OK

Evaluation on Malicious SSH Clients

Malware Sample	Closest Legitimates	GED	Similarity
Abafar_Client	ssh-6.0	12	74.370
	ssh-7.2p2	13	81.606
Akiva_Client_2	ssh_debian_stretch	82	54.056
	ssh-6.6	86	53.958
Atollon_Client_2	ssh-7.6	17	76.236
	ssh-7.9	21	58.765
Bespin_Client	ssh-6.3	19	76.344
	ssh_debian_bullseye	20	67.825
Crait_Client	ssh-6.0	13	67.044
	ssh-6.1	13	69.133
	ssh-7.1	13	67.027
Chandrila_Client	ssh-9.0	46	36.845
	ssh_debian_bookworm	46	36.846
Endor_Client	ssh_debian_stretch	90	41.872
	ssh-7.3	96	38.149
Endor_Client_5	ssh_debian_stretch	90	41.872
	ssh-7.3	96	38.149
Mimban_Client_2	ssh-6.4	12	68.156
	ssh-6.7	14	66.100
Mimban_Client_3	ssh-6.4	12	56.527
	ssh-6.0	20	64.825
Onderon_Client_2	ssh-7.1	10	63.487
	ssh-6.8	13	67.732
PolisMassa_Client	ssh-6.4	12	61.655
	ssh-6.7	16	54.201
PolisMassa_Client_2	ssh-7.1	10	74.404
	ssh-6.7	11	71.111
Ebury_Injected_Client	ssh-6.8	20	80.458
	ssh-ubuntu16.04	21	84.277
	ssh-ubuntu18.04	–	85.224

- Samples from Eset*.
- We only use our way of finding the "original" program.
- Results deviate in the two "bad" classes, some modifications more blatant than others.

* https://www.welivesecurity.com/wp-content/uploads/2018/12/ESET-The_Dark_Side_of_the_ForSSHe.pdf

PoC evaluation Under Different Interpretations

- We shrink the database of legitimate programs (remove each with a probability of 0.3).

- Test all malicious samples and the removed legitimate ones against the “crippled” database.

PoC evaluation Under Different Interpretations

“Liberal” (0-70 bad, 70+ OK)

	Detected	Not Detected
Legitimate	1 (FP)	14 (TN)
Modified	10 (TP)	3 (FN)

Accuracy	0.857
Precision	0.909
Recall	0.769

“Conservative” (0-90 bad, 90+ OK)

	Detected	Not Detected
Legitimate	6 (FP)	9 (TN)
Modified	13 (TP)	0 (FN)

Accuracy	0.785
Precision	0.722
Recall	1.000

Conclusions

- Approach is viable.
- Tool alerts the analysts, graph representation of deviations is appropriate for visualization.
- Careful with the choice of tracing tools.

- Further analysis of our methods outputs?

Time for Your Questions!