Challenges of collaborative malware analysis

Polichombr

S. Le Berre A. Chevalier T. Pourcelot

ANSSI/COSSI/DTO/BFS — SOGETI ESEC

SSTIC — Rennes — June 1, 2016

Plan

What is it about

Operational malware analysis

- Malwares everywhere!
- Malware writers are more numerous than malware reversers
- Let's work as a team to tackle them!

Plan

Goals

Why reverse malwares?

- Technical follow up on adversary tools
  - Many adversaries, many tools
- Sample identification
  - More effective incident response! . . .
- Produce detection elements
- Capitalization of experience
- Threat intelligence & know your adversary

Formalization

SOGETI

Inputs

- Samples
- Context, associated documents, detection rules, . . .

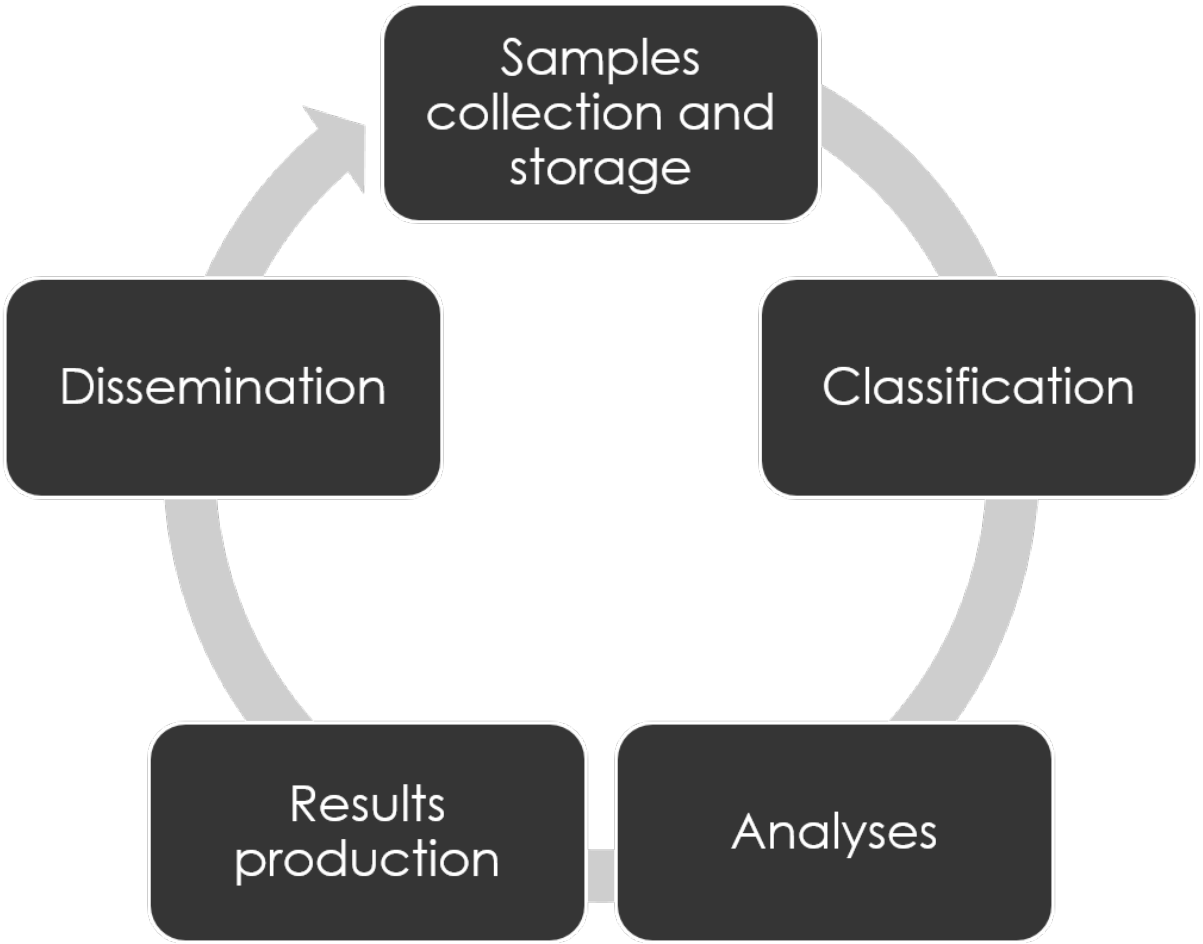
Output

- IOC and threat reports
- Adversary toolset knowledge

Constraints

- DO IT QUICK!
- Don’t waste time
- Don’t forget anything
- Limited manpower

Analysis cycle



Storage and collection

Challenges

- Collection
- Volume (many adversaries, many tools, many versions of these tools)

Effective storage needs

- Browsable (metadata)
- Usable

Problems

- Filer storage
- Storage on reverser's laptop or drives

Classification

Benefits

- Family identification
- Identification of similarities
- Sample triaging

Current techniques

- `Yara` and dynamic execution signatures
- *Mandiant*'s `imphash`
- `Control Flow Graph` comparison
- Metadata comparison

SOGETI

Analysis

Benefits

- Answer technical questions about the sample
- Identify interesting points in the binary

Methods

- Top-down: start from entry points
- Bottom-up: start from IAT or patterns

Challenges

- Automated analysis: fast but incomplete
- Manual analysis : time consuming, prone to omissions
- Team work: whiteboards and meetings are not sufficient

Results production and capitalization

Sample information

- Raw technical information
- Techniques used
- Code overview

Family information

- Overview: sophistication, variants, etc
- Detection techniques
- Tools (unpacking scripts, etc.)

Problems

- Lost reports, IDB corruption, ...

Dissemination and feedback

Benefits

- Propagation on existing dataset,
- Information shared: improved detection, actors knowledge, ...
- Information gained: new samples, technical/context feedback, ...

Challenges

- Multiple types of interlocutors = multiple types of languages and channels
- Effective technical information sharing
- Both external (sensitivity) AND internal (experience)

Automation

Plan

POLICHOMBR

Why this new tool?

History

- Tool developped by BFS in 2014
- Originally Ruby/PHP/Python for *Windows* (yes. . .)
- Evolving since ;)

Addressed challenges

- Storage!
- Information/Knowledge centralization
- Collaborative teamwork
- Automation
- Classification (introducing the `MACHOC` algorithm)

Bricks

WebUI

- Macro overview
- Expose an API

Analysis engine

- Run all the things!

Disassembly engine

- METASM

User's endpoint

- IDA Python script

Datatypes

Binaries

- PE/ELF/Shellcodes/...
- Associated metadata

Families

- Store contexts, utilities, overview information
- Tree used to organize samples/threats

Signatures

- Machoc
- Yara

Binary classification

Problems

- `MD5`, `SHA*` not adapted (by definition)
- `SSDEEP`, `SDHash` not adapted to executables

Goals

- Act like a fingerprint of the program
- Lightweight (can be exchanged by mail)
- Resistant to recompilation
- Resistant to architecture change (`x86_64`)

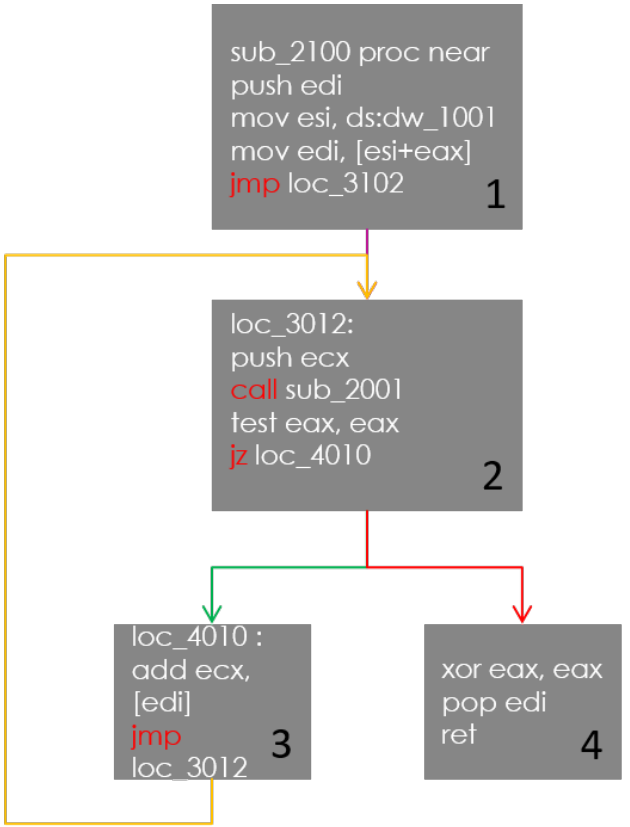
Machoc algorithm

In a nutshell

Control Flow Graph "snapshot" of a function

Algorithm

- Blocks and call labelling
- Translate to text
- → `1:2;2:c,3,4;3:2;4:;`
- Murmurhash3
- → `0x94167eb0`
- For each function in sample, concatenate



Usages

Sample classification

- Threshold = 80% (empiric)

Information propagation

- Between samples
- Propagate all the names!

Analyzing a new sample

Submission

WebUI, API or directly from IDA

Automated analysis: plugins

- Metadata, strings, machoc extraction
- Add comments, renames, hints
- Output a brief text summary

Classification

- Strong/automated identification: `Yara` (extended with `Machoc`)
- Soft/suggested identification: `imphash`, `Machoc_80`

Results storage

Sample documentation

- Analysts notes
- Checklist
- IDA actions

Family documentation

- Analysts notes
- Detection items (SNORT rules, OpenIOC, etc.)
- Classification signatures (`Yara`, `Machoc`)
- Other elements: context, reports, tools
- Analysts
- Etc.

Data export

For analysts: `Machex`

- Can include any information about the sample
- Specifically information about functions, names and machoc hashes
- Can be imported back

For consumers

- Reports, detection rules, IOC, samples archive
- Sensitivity management

For tools

- Expose all the data with an API

Team reversing

SOGETI

Skelenox

- IDA Python script
- Synchronization between user's IDA database and Polichombr
- Push/pull changes (including other user's)
- Names, comments, types, ...
- Realtime identification (using Machoc hashes)

Plan

DEMO DEMO DEMO

SOGETI

Automated analysis

- Sample metadata
- Classification
- Automated reverse!

Bonus

- OpenIOC Export

Plan

Conclusion

What we try to achieve

- Quickly and efficiently produce information about malwares
- Provide a tool for automation and communication of analyses

About the tool

- `https://github.com/ANSSI-FR/polichombr`
- Can be used for other collaborative reversing tasks =)
- Pull requests, feedback and suggestions are welcome!

HR

- If you like malware analysis,
- If you were not lost in this presentation,
- BFS & Sogeti are hiring! ;-)

Q&A

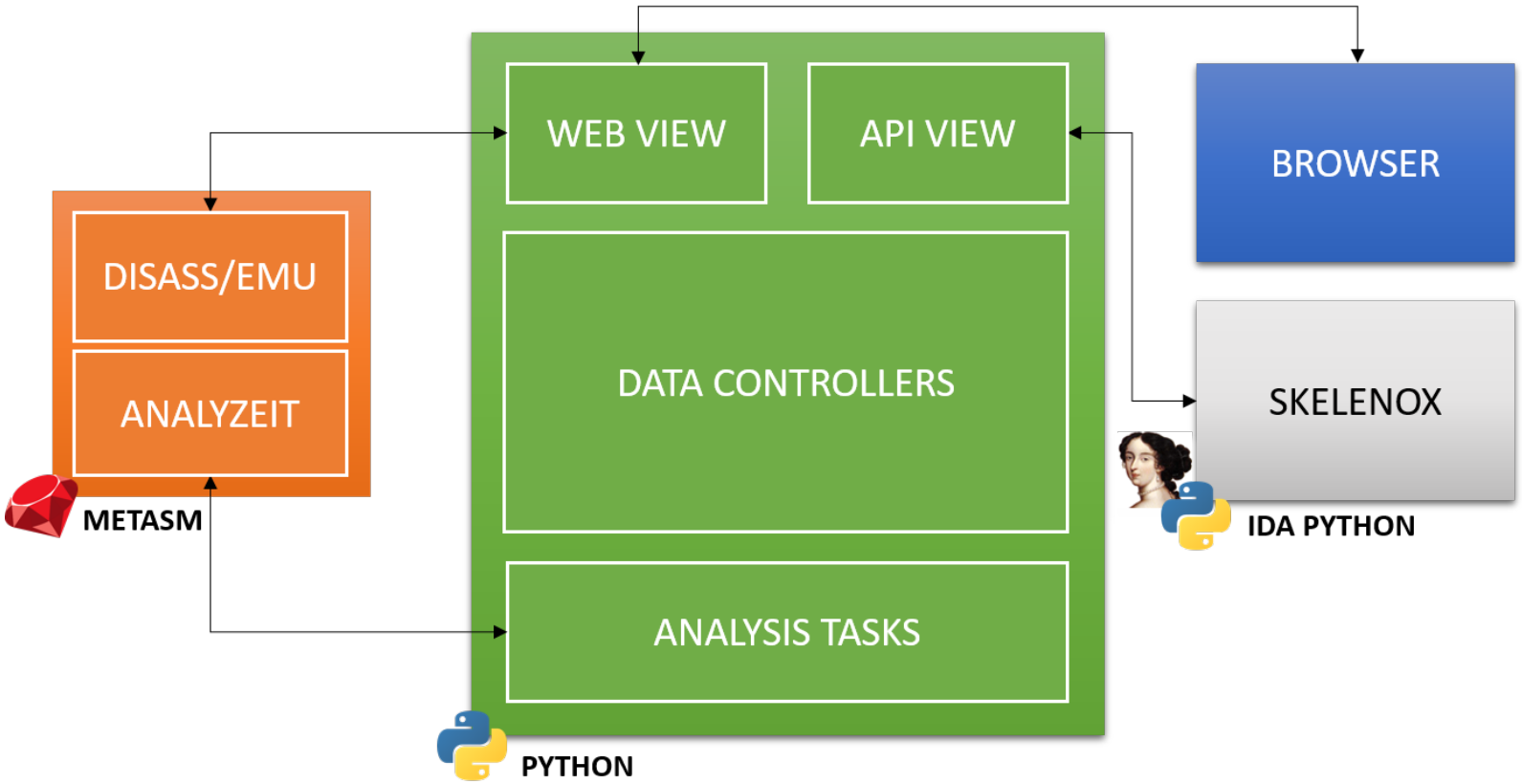
Thank you for your attention!

Questions?

Plan

6 Backup

Architecture schema



TODO

SOGETI

Missing features

- Overlay/resources extraction
- Emulator
- PDB generation
- More tasks!
- More IDA functionalities (structs, segments, . . .)
- Fix bugs!

Existing tools and limits

Main tools:

- IDAScope
- IDAToolbag
- Viper
- CrowdRE
- Manalyze

Why we didn't choose them for the task

- Often unmaintained
- Or not open source
- Scaling problem
- None of them were a silver bullet for our problems