Binary Sight-Seeing: Accelerating Reverse Engineering via Point-of-Interest-Beacons

August See, Maximilian Gehring,
Mathias Fischer, Shankar Karuppayah

Motivation

- Reverse Engineering
  - Time consuming
  - Driven by experience and beacons
    - E.g., function names and strings
- Goal: Provide More Useful Beacons
  - Instructions that interact with specific data
  - Example: Where is the message of type 0xcafe constructed?

Identify salient points of interest (POIs) within binary code

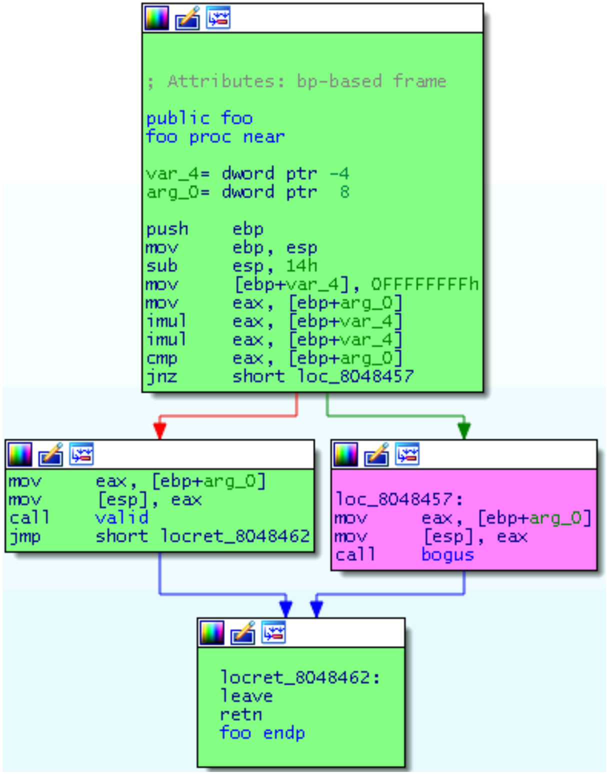
Filter the POIs on their relevance

Evaluation on ransomware and P2P botnets

Automated botnet monitoring (peer list extraction and crawling)

Related Work

- Taint Analysis [1]
  - Method for tracking how data interacts with a program
  - Mark initial data then mark all instructions that access this data
  - Complex propagation calculation
    - E.g., data is written and then read from a file
- Automatic Botnet Monitoring [2]
  - Reusing existing functionality
  - Replay C&C messages and monitoring responses



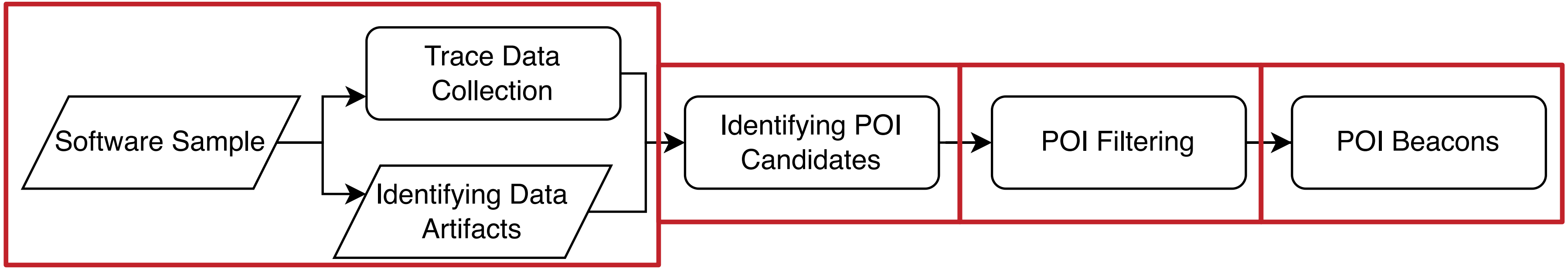
Core Idea for Finding POI

- Identification of Points of Interest (POI)
  - Locations in a binary which the analyst is interested in

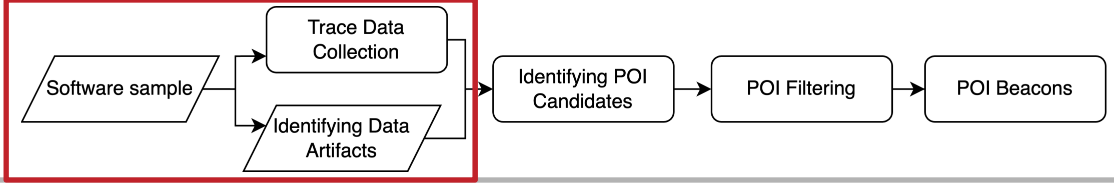
- POIs are Dependent on Data
  - ”Points where the binary interacts with data X”
    - Encryption function ⇒ Location where cleartext and ciphertext content is accessed
  - Data is often observable or known
    - Traffic, file content, …

```
// eax, address of the message buffer
mov [eax], 0xcafe
// further message construction
push eax
// send message buffer to recipient
call send_message
```

Overview of Using our Approach on a Binary



Preparation

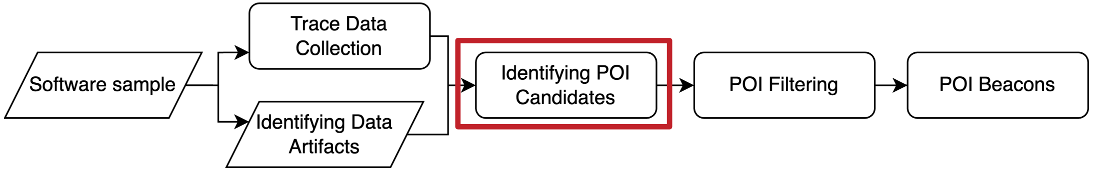


- Identifying Data Artifacts via Monitoring Software
  - Any.run, Cuckoo, Wireshark, Procmon, …
  - Or prior knowledge

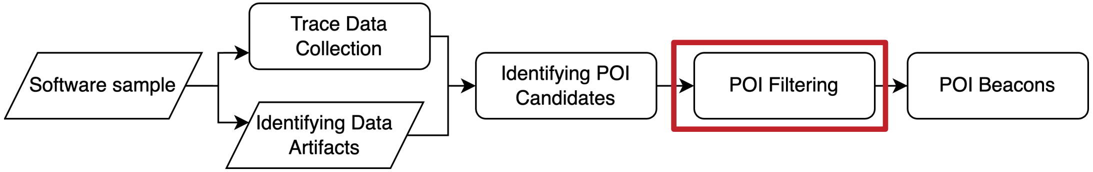
1804	12.288001	192.168.178.140	192.168.178.255	UDP	63	35205 → 32414 Len=21
1805	12.288003	192.168.178.140	192.168.178.255	UDP	63	53241 → 32412 Len=21
1806	12.297125	192.168.178.129	185.70.42.43	TLSv1…	311	Application Data
1807	12.299523	192.168.178.129	185.70.42.55	TCP	78	62023 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3309453112 TSecr=0 SACK_PERM
1808	12.321176	185.70.42.43	192.168.178.129	TCP	66	443 → 61287 [ACK] Seq=13737 Ack=441 Win=1022 Len=0 TSval=2402301247 TSecr=3132671460
1809	12.401501	185.70.42.43	192.168.178.129	TCP	1294	443 → 61287 [ACK] Seq=13737 Ack=441 Win=1022 Len=1228 TSval=2402301328 TSecr=3132671460 [T
1810	12.401504	185.70.42.43	192.168.178.129	TLSv1…	322	Application Data
1811	12.401781	192.168.178.129	185.70.42.43	TCP	66	61287 → 443 [ACK] Seq=441 Ack=15221 Win=2024 Len=0 TSval=3132671564 TSecr=2402301328
1812	12.697849	192.168.178.70	255.255.255.255	UDP	214	59727 → 6667 Len=172

- Trace Data Collection
  - Run our tool to get the execution trace of the program
  - Trigger wanted behavior

Identifying POI Candidates



- Determine whether an instruction interacted with the data artifacts
- Single Instruction POIs: Complete data artifact is accessed in one instruction
  - Check accessed memory and registers
    - `mov eax, 0xcafe` [Register POI]
    - `push eax` [Memory POI]
- Multi Instruction POIs
  - Complicated as data might be accessed in random order
    - Hello World
  - Solution: Search surrounding memory area when a searched byte is written
    - Optimize by checking memory allocations



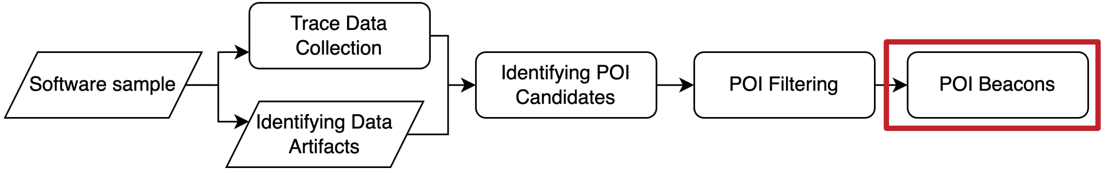
Filtering POIs

- Found POIs might be noisy
  - Data propagates from function to function
  - Not every function is of interest
    - `readfile` → `memcopy` → `encrypt` → `memcopy` → `writefile`

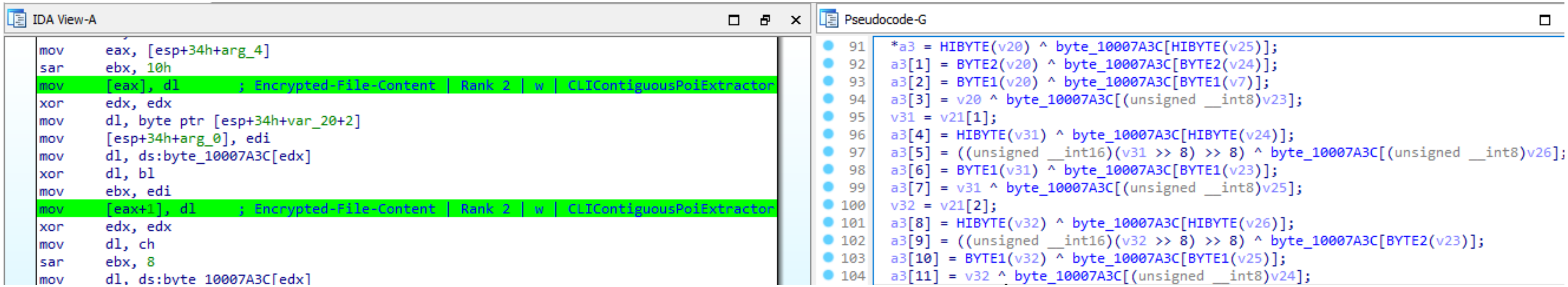
- New Metric: Confidence Score
  - Calculate how exclusive a function operates on our data
  - D := set of data artifacts for identifying POIs
  - Single Instruction POI
    - Determine the ratio of accessed data in D to the total accessed data
  - Multi Instruction POI
    - Calculate the ratio of accessed bytes in D to the total accessed bytes
  - The more complete D is, the higher the scores

$$scores_S(p) := \sum_{d \in D} \frac{C(d, \text{trace}(p))}{|\text{trace}(p)|}$$

POI Beacons
Plugin for IDA



POI ≔ Address, confidence score, list of accessed patterns



Type	Location	Pass count	Hardware	Condition	Actions	State	Comment	Group
Abs	0x100069DA (sub_10006940:loc_100069DA)				Break	Disabled	Cleartext-File-Content | Rank 1 | r | CLIContiguousPoiExtractor	Default
Abs	0x100069DD (sub_10006940+9D)				Break	Disabled	Encrypted-File-Content | Rank 1 | r | CLIContiguousPoiExtractor	Default
Abs	0x10006A0D (sub_10006940+CD)				Break	Disabled	Encrypted-File-Content | Rank 1 | r/w | CLIContiguousPoiExtractor	Default
Abs	0x100064CC (sub_10006280+24C)				Break	Disabled	Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor	Default
Abs	0x100064E2 (sub_10006280+262)				Break	Disabled	Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor	Default
Abs	0x100064F8 (sub_10006280+278)				Break	Disabled	Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor	Default
Abs	0x1000650D (sub_10006280+28D)				Break	Disabled	Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor	Default

Evaluation – Confidence Score

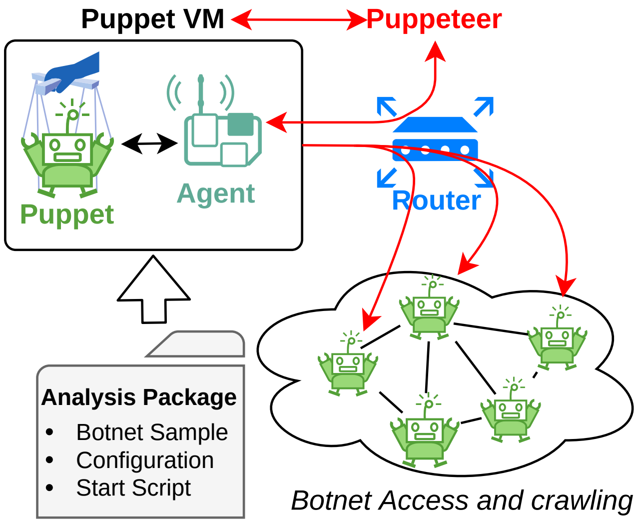
- P2P Botnets
  - No central C&C structure but peer lists
    - $P_{bootstrap}$ ≔ Initial peer list
  - Exchange of peers

- Show that the found POIs are of high quality
  - How well does the confidence score predict the quality of extracted IPs
  - Use them to automatically monitor P2P botnets without human intervention

- Objective: Find the instruction that interacts with the peer list
  - Then we can extract peers by monitoring this instruction
  - Input Data: IPs obtained by monitoring network connections ($P_{bootstrap}$)

Evaluation – Setup

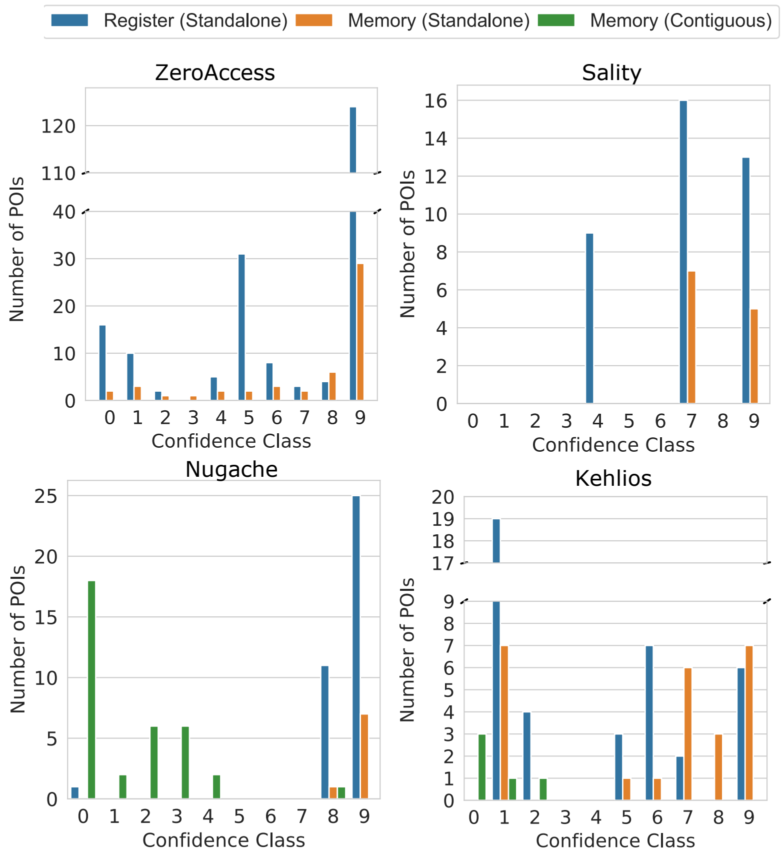
- Local P2P Botnet
  - Infect 40 VMs with the botnet
  - Let them find each other in the local network
    - P_{local} ≔ Available local peers
    - Peer list of a bot should only contain $P_{bootstrap} \cup P_{local}$
  - Observed botnets: Sality, Nugache, ZeroAccess, Kelihos



- Use each found POI to extract new peers (P_{local})
  - Peer can be
    - from $P_{bootstrap}$
    - from P_{local} (correct peers)
    - not from either → wrong

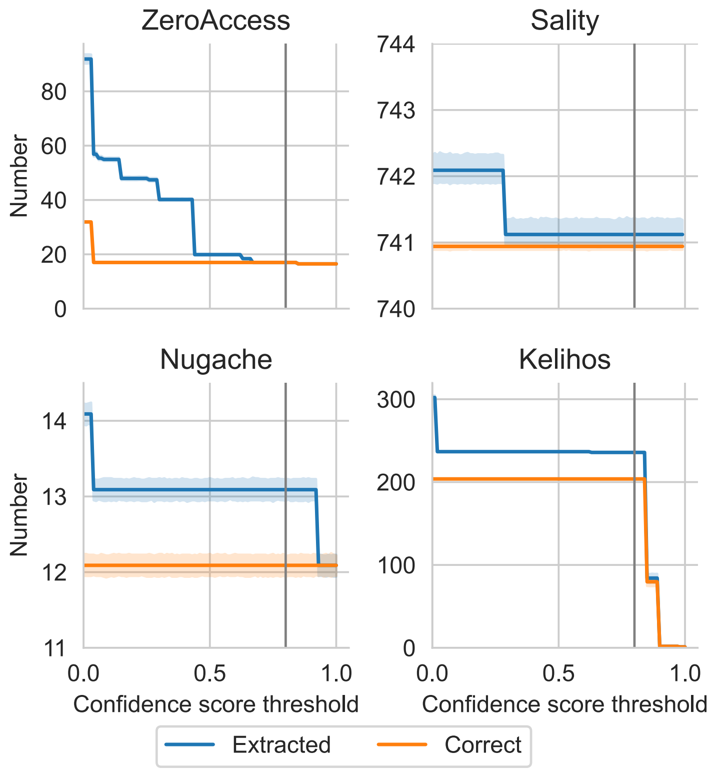
Evaluation – Found POIs

- For each Botnet multiple POIs could be found
  - Different instruction address
- High Confidence Score for Standalone POIs
  - IPs can fit in 4 Bytes
- Low confidence score POIs are low quality?
  - Cannot extract valid peers?



Evaluation – Confidence Score

- Use each found POI to extract new peers (P_{local})
  - from $P_{bootstrap}$
  - from P_{local} (correct peers)
  - not from either → wrong
- Cummulative Plot
  - We plot for confidence x the number of POIs for which the confidence score is ≥ x
  - Average of extracting peers for one membership cycle
- Amount of wrong peers decreases
  - Confidence score predicts quality of POIs
- Limitation: POIs are susceptible to certain obfuscation approaches



Conclusion

- Approach to provide more beacons for reverse engineering
- Evaluated on P2P botnets and ransomware
  - Find encryption function of ransomware
  - Find peer list functions in P2P botnets
- Approach provides high quality beacons
  - They can be used in automated programs
  - They can be used for manual reverse engineering
    - IDA and Ghidra plugins available

Thanks for your attention!

Questions?
Thoughts?
Ideas?

August See
Universität Hamburg
Computer Networks (NET)
richard.august.see@uni-hamburg.de

Bibliography

- [1] Edward Schwartz, Thanassis Avgerinos, and David Brumley. 2010. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). Proceedings - IEEE Security and Privacy, 317–331. https://doi.org/10.1109/SP.2010.26
- [2] lemens Kolbitsch, Thorsten Holz, Christopher Kruegel, and Engin Kirda. 2010. Inspector gadget: Automated extraction of proprietary gadgets from malware binaries. In 2010 IEEE Security and Privacy. IEEE, 29–44.