INTRODUCTION

Security Researchers in

Mariano Graziano
Jonas Zaddach
LET'S TALK ABOUT THE THREAT LANDSCAPE
THREAT LANDSCAPE

1.5 MILLION Malware Samples DAILY = 10,000
AV PIPELINE OVERVIEW

- SAMPLES
- AV SCAN
- SANDBOX ANALYSIS

MALWARE
MALWARE (NEEDS SIGNATURES)
NOT MALICIOUS
MALWARE DETECTION CHALLENGE

≈ 560,000 signatures over a 3-month period

≈ 9,500 Signatures DAILY

→ Huge number of signatures
→ Pattern-based signatures can reduce resource footprint compared to hash-based signatures
BASS OVERVIEW

MALWARE CLUSTER ➔ INSPECT UNPACK ➔ FILTER ➔ GENERATE SIGNATURE ➔ EVALUATE ➔ AV SIGNATURE
• Clustering is NOT a part of BASS!
• Several cluster sources feed BASS
  – Sandbox Indicator of Compromise (IoC) clustering
  – Structural hashing
  – Spam campaign dataset
UNPACKING & INSPECTION

• Extract all content ClamAV can extract
  – ZIP archives
  – Email attachments
  – Packed executables
  – Nested documents: e.g., PE file inside a Word document
  – …

• Gather information about file content
  – File size
  – Mime type/Magic string
  – …
FILTERING

• Reject clusters with wrong file types
  – In the near future BASS will handle any executable file type handled by the disassembler (IDA Pro)
  – Currently limited to PE executables

• Clean outliers with wrong file types from clusters
DISASSEMBLING

- Export disassembly database
- Currently uses IDA Pro as a disassembler
  - Others are possible in the future
FINDING COMMON CODE

- Use binary diffing to identify similar functions across binaries
- Build similarity graph between functions and extract largest connected subgraph
FINDING COMMON CODE

- Test found function against a database of whitelisted functions
  - Kam1n0, a database for binary code clone search, contains functions of whitelisted samples
  - If a found function is whitelisted, take the next-best subgraph

Kam1n0
FINDING AN LCS

• Use k-LCS algorithm to find a longest common subsequence

DIFFERENCE BETWEEN LONGEST COMMON SUBSTRING AND LONGEST COMMON SUBSEQUENCE

<table>
<thead>
<tr>
<th>LONGEST COMMON SUBSTRING</th>
<th>LONGEST COMMON SUBSEQUENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBACABACCBCBA</td>
<td>ABBACABACCBCBA</td>
</tr>
<tr>
<td>ACBCCBACCACB</td>
<td>ACBCCBACCACB</td>
</tr>
<tr>
<td>BACCABBBBBBAC</td>
<td>BACCABBBBBBAC</td>
</tr>
</tbody>
</table>

- -> BACC
  substring appears verbatim in all strings

- -> ABBAC
  subsequence appears in the same order in all strings, other characters can be inserted between

• Implemented Hamming-kLCS described by C. Blichmann [1]
Hamming distance between all strings is computed

2-LCS algorithm (Hirschberg algorithm) is applied to strings with lowest distance

Resulting LCS is kept $\rightarrow$ Rinse and repeat

ABBACABACCBCA

ACBCBACCACB

BACCABBBBBBBAC
FINDING AN LCS

- Hamming distance between all strings is computed
- 2-LCS algorithm (Hirschberg algorithm) is applied to strings with lowest distance
- Resulting LCS is kept → Rinse and repeat

ABBACABACCBACA
ACBCBACCACB
BACCABBBBBBBAC

8
12
11
FINDING AN LCS

- - - - - - - - - - - - - - - - - - - - -

• Hamming distance between all strings is computed
• 2-LCS algorithm (Hirschberg algorithm) is applied to strings with lowest distance
• Resulting LCS is kept → Rinse and repeat

ABBACABACCCBCA
ACBCBACCACB
BACCABBBBBBAC

ABBACCCB
FINDING AN LCS

- Hamming distance between all strings is computed
- 2-LCS algorithm (Hirschberg algorithm) is applied to strings with lowest distance
- Resulting LCS is kept → Rinse and repeat

ABBACCB

BACCABBBBBBBBAC
FINDING AN LCS

- Hamming distance between all strings is computed
- 2-LCS algorithm (Hirschberg algorithm) is applied to strings with lowest distance
- Resulting LCS is kept → Rinse and repeat
GENERATING A SIGNATURE

- Create ClamAV signature
  - Find possible “gaps” in result sequence
  - Delete single characters
- Find a common name
  - Use AvClass to label cluster

**SIGNATURE:** Win.Trojan.Example:0:*cafebabe*dead*beef

**ORIGINAL FILES:**

1. ca fe ba be 31 de ad 35 37 be ef 38 |....1..57..8|
2. 31 ca fe ba be de ad be ef 35 38 37 |1........587|
3. 30 ca fe ba be 31 de ad 37 be ef 38 |0.....1..7..8|
VALIDATION

- False Positive testing
  - Against a set of known clean binaries
- Manual validation by Analyst
  - Assisted by CASC plugin [4]
  - Matched binary parts are highlighted in IDA Pro
CONCLUSION
LIMITATIONS

• Only works for executables
• Does not work well for
  – File infectors (Small, varying snippets of malicious code)
  – Backdoors (Clean functions mixed with malicious ones)
• Alpha stage
CONCLUSION

• Presented automated signature generation system for executables
• Implemented research ideas not available as code
  – VxClass from Zynamics
• Code will be available open-source
  – For others to try, improve and comment on

https://github.com/CISCO-TALOS/bass
1. “Automatisierte Signaturgenerierung für Malware-Stämme”, Christian Blichmann
2. “AVClass: A Tool for Massive Malware Labeling”, Sebastian et al.,
3. “Kam1n0: MapReduce-based Assembly Clone Search for Reverse Engineering”, Ding et al.,
4. CASC IDA Pro plugin, https://github.com/Cisco-Talos/CASC
5. VxClass – Automated classification of malware and trojans into families
   https://www.zynamics.com/vxclass.html