# A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion:
## PC, Mobile, and Web

Alexei Bulazel & Bülent Yener
River Loop Security
Rensselaer Polytechnic Institute (RPI)

RENSSELAER POLYTECHNIC INSTITUTE
KNOWLEDGE AND THOROUGHNESS
• 1824 •

DEEPSEC

# Introduction

- Automated dynamic malware analysis is essential to keep up with modern malware (and potentially malicious software)

- Problem: malware can detect and evade analysis

- Solution: detect or mitigate anti-analysis

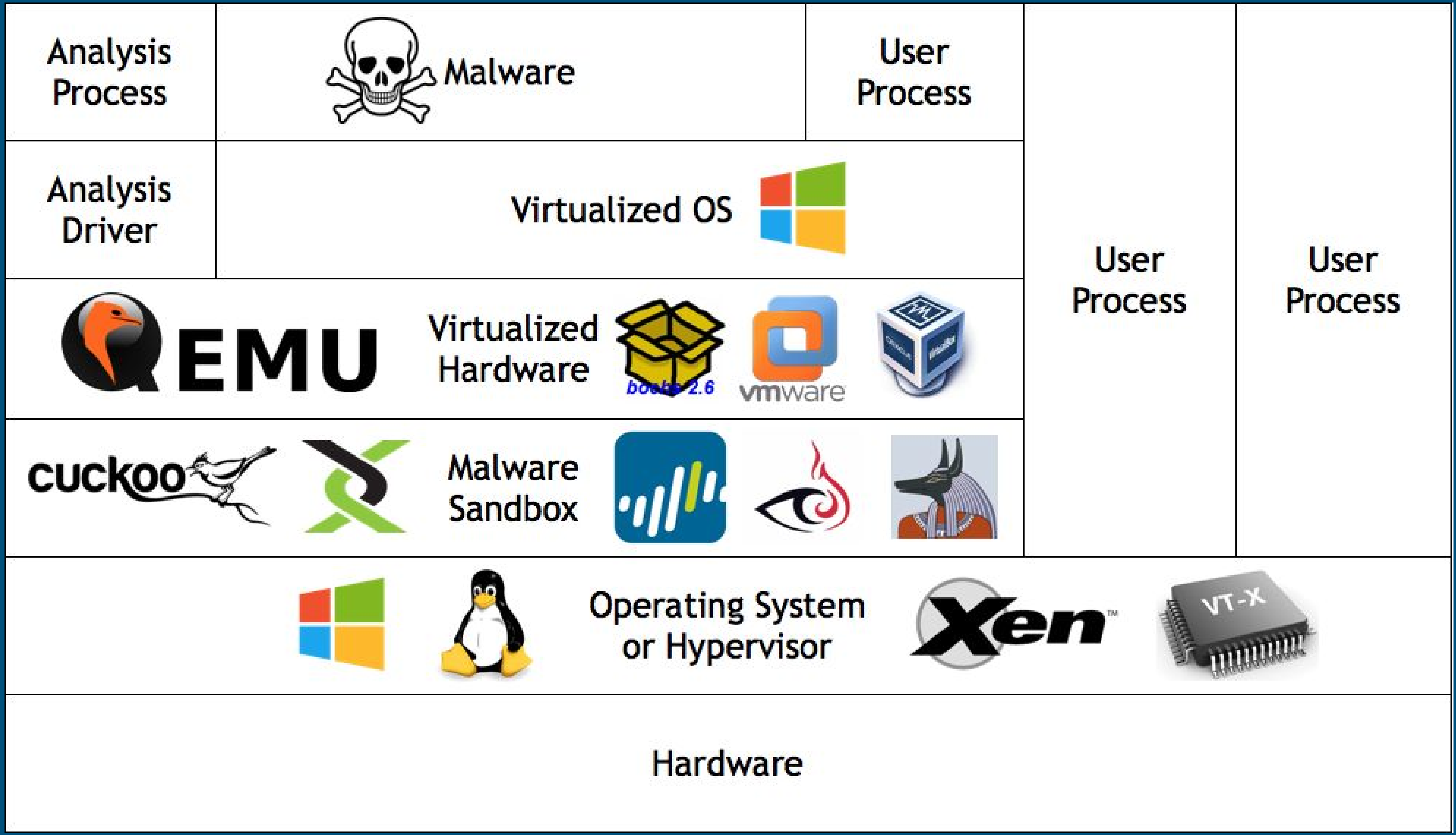
# Scope

- Survey of ~200 works on evasive malware techniques, detection, mitigation, and case studies
- Mostly academic works, with a few industry talks and publications

- In this presentation - focus on PC-based malware experimentation, more discussion than survey

# Dynamic Automated Analysis Systems

a.k.a:
“malware sandboxes”
“detonation chambers”



# Takeaways

- Evasive malware and defenders continually evolve to counter one another
- The fight between malware and analysis systems is likely to continue long into the future
- There are immense challenges to experimental evaluation and the ability to establish ground truth

# Presentation Outline

1. Introduction
2. Offense - Detecting Analysis Systems
3. Defense - Detecting Malware Evasion
4. Defense - Mitigating Malware Evasion
5. Discussion
6. Conclusion

# Offense - Detecting Analysis Systems

- Fingerprint Classes
  - Environmental Artifacts
  - Timing
  - CPU Virtualization
  - Process Introspection
  - Reverse Turing Tests
  - Network Artifacts
  - Mobile Sensors
  - Browser Specific

```
bool beingAnalyzed = DetectAnalysis();

if(beingAnalyzed)
{
    BehaveBenignly();
}
else
{
    InfectSystem();
}
```

# Environmental Artifacts & Timing


- Unique distinguishing characteristics of the analysis environment itself
  - Usernames
  - System settings
  - Date
  - Installed software
  - Files on disk
  - Running processes
  - Number of CPUs
  - Amount of RAM
- Timing discrepancies in analysis systems
- Sources:
  - Emulation / virtualization overhead
  - Analysis instrumentation overhead
  - Overhead of physical hardware instrumentation (potentially)
- Challenging to mitigate
  - Garfinkle et al: “extreme engineering hardship and huge runtime overhead”

# CPU Virtualization & Process Introspection

- CPU "Red Pills"
- Discrepancies in CPU behavior introduced by virtualization
  - Erroneously accepted/rejected instructions
  - Incorrect exception behavior
  - Flag edge cases
  - MSRs
  - CPUID/SIDT/SGDT/etc discrepancy
- Particularly applicable for emulators
- Discrepancies in internal state
  - Memory or register contents
  - Function hooks
  - Injected libraries
  - Page permission eccentricities
- Commonly used in anti-DBI

# Reverse Turing Tests & Network Artifacts

- *Computer* decides if *it* is interacting with computer or human
- Passive: mouse movement, typing cadence, process churn, scrolling
- Active: user must click a dialogue box
- Wear-and-Tear: evidence of human use, copy-paste clipboard, “recently opened” file lists, web history, phone camera photos

- Fixed IP address
- Network isolation
- Incorrectly emulated network devices or protocols
- Unusually fast internet service

# Detection - Discussion

- Variety of sources: underlying technologies facilitating analysis, system configuration, analysis instrumentation

- Easy to use = easy to mitigate
- Difficult to use = difficult to mitigate

- Reverse Turing Tests seem to be growing in relevance, and are extremely difficult to mitigate against

# Presentation Outline

1. Introduction
2. Offense - Detecting Analysis Systems
3. Defense - Detecting Malware Evasion
4. Defense - Mitigating Malware Evasion
5. Discussion
6. Conclusion

# Detecting Malware Evasion

- Detecting that malware exhibits evasive behavior under dynamic analysis, but not mitigating evasion
  - Comparatively fewer works relative to mitigation work
- Early work - detecting known anti-analysis-techniques
  - 2008: Lau et al.'s DSD-Tracer
- Most works use multi-system execution
  - Run malware in multiple systems and compare behavior offline - discrepancies may indicate evasion in one or more of these systems

# Multi-System Execution

- Instruction-level (2009: Kang et al.)
  - Too low level, prone to detect spurious differences
- System call-level (2010: Balzarotti et al. / 2015: Kirat & Vigna - MalGene)
  - Higher level than just instructions
  - MalGene uses algorithms taken from bioinformatics work in protein alignment
- Persistent changes to system state (2011: Lindorfer et al. - Disarm)
  - Jaccard distance-based comparisons
- Behavioral profiling (2014: Kirat et al. - BareCloud)
  - *What* malware did vs. *how* it did it, "hierarchical similarity" algorithms from computer vision and text similarity research

# Evasion Detection - Discussion

- Multi-system execution is a common solution for evasion detection
- Offline algorithms do not detect evasion in real time
- Evolution over time to increasingly complex algorithmic approaches, working over increasingly abstracted execution traces
- Detection does not solve the main challenge of evasion, so there is less work in the field compared to mitigation research

# Presentation Outline

1. Introduction
2. Offense - Detecting Analysis Systems
3. Defense - Detecting Malware Evasion
4. Defense - Mitigating Malware Evasion
5. Discussion
6. Conclusion

# Defense - Mitigating Evasion

- Mitigating evasive behavior in malware so that analysis can proceed unhindered

- Early approaches
  - Binary Modification
  - Hiding Environment Artifacts
  - State Modification
  - Multi-Platform Record and Replay
- Path Exploration
- Hypervisor-based Analysis
- Bare Metal Analysis & SMM-based Analysis
- Discussion

# Early Approaches

- Binary Modification
  - 2006: Vasudevan et al. - Cobra
  - Emulate code in blocks like QEMU
    - Remove or rewrite malware instructions that could be used for detection
- Hiding Environmental Artifacts
  - 2007: Willems et al. - CWSandbox
    - In system kernel driver hides environmental artifacts
  - Oberheide later demonstrated several detection techniques against CWSandbox
- State Modification
  - 2009: Kang et al.
    - Builds upon detection work
    - “dynamic state modification” (DSM), modifications to state force malware execution down alternative paths
- Multi-Platform Record and Replay
  - 2012: Yan et al. - V2E
    - Kang et al.’s DSMs are not scalable for multiple anti-analysis checks
    - Don’t mitigate individual occurrences of evasion, make evasion irrelevant because systems are inherently transparent

# Path Exploration

- 2007: Moser et al.
  - Looks broadly at code coverage and analyzing trigger-based malware
  - Track when input is used to make control flow decisions, change it to force execution down different code paths
- 2008: Brumley et al. - MineSweeper
  - Trigger-based malware focused
  - Represents inputs to potential triggers symbolically, while other code is executed concretely

# Hypervisor-based Analysis

- 2008: Dinaburg et al. - Ether
  - Catch system calls and context switches from Xen
  - Despite extensive efforts to make analysis transparent, Pék et al. created nEther and were able to detect Ether

- 2009: Nguyen et al. - MAVMM
  - AMD SVM with custom hypervisor
  - Thompson et al. subsequently demonstrated timing attacks that can be used to detect MAVMM and other hypervisor based systems

- 2014: Lengyel et al. - DRAKVUF
  - Xen-based, instruments code with injected breakpoints

# Bare Metal Analysis

- 2011, 2014: Kirat et al. - BareBox & BareCloud
  - BareBox - in-system kernel driver
  - BareCloud - post-run disk forensics
- 2012: Willems et al.
  - Hardware-based branch tracing features
  - Analyzed evasive PDFs
- 2016: Spensky et al. - LO-PHI
  - Instrument physical hardware
  - Capture RAM and disk activity at the hardware level
  - Scriptable user keyboard/mouse interaction with USB-connected Arduinos
- SMM-based analysis: all the transparency benefits of bare metal, while restoring introspection
  - Full access to system memory, protection from modification, high speed, protection from introspection
- 2013 & 2015: Zhang et al. - Spectre, MalT
  - Spectre: SMM-based analysis, 100x faster than VMM based introspection
  - MalT - SMM-based *debugging*
- 2016: Leach et al. - Hops
  - SMM memory snapshotting and PCI-based instrumentation

# Mitigation - Discussion

- Two broad categories: active and passive mitigation
  - Active - detect-then-mitigate
  - Passive - build inherent transparency
- Passive approaches have been more prevalent
  - Hypervisors, bare metal, etc
- Bare metal is the cutting edge in academic research, but it may not be scalable to industry applications
  - Promising, but not a panacea against any class of attacks other than CPU-based

# Presentation Outline

1. Introduction
2. Offense - Detecting Analysis Systems
3. Defense - Detecting Malware Evasion
4. Defense - Mitigating Malware Evasion
5. Discussion
6. Conclusion

# Discussion

- Offensive Research
    - Reverse Turing Tests
    - Detecting Bare Metal Analysis
- Defensive Research
    - Improving Bare Metal Analysis
    - Heuristic Evasion Detection
    - Passing Reverse Turing Tests
- Game Theory Formalizations
- Research Evaluation
    - Establishing Ground Truth
    - Challenges in research evaluation
    - Suggestions for Improvement

# Offensive Research

- Reverse Turing Tests
  - Difficult to mitigate against
  - Increasingly relevant as analysis systems become transparent
  - Look to anti-cheating research for online gaming
- Detecting bare metal analysis
  - Still vulnerable to everything except CPU-based attacks
  - Look to detecting analysis instrumentation

# Defense - Improving Bare Metal Analysis

- Improving bare metal analysis - efficient, introspection, and stalling mitigation
  - Efficiency
    - 2016: Vadrevu and Perdisci - MAXS - improve efficiency by 50% on average with less than 0.3% information loss in analysis
  - Introspection
    - SMM needs further research
  - Stalling mitigation
    - Difficult to mitigate against with current bare metal systems
    - Performance tracing technologies may provide a direction forward

# Defense - Heuristic Evasion Detection

- Can the behaviors involved in evasion before conditional branching occurs be detected heuristically?

- Inspirations
    - Code *fragility* may indicate maliciousness
    - Heuristic detection in enterprise and personal AV/endpoint products
    - Stalling detection techniques
    - Anti-anti-DBI heuristics

# Defense - Passing Reverse Turing Tests

- Believably simulating human presence as reverse Turing Tests become more prevalent
- Inspirations:
  - UNVEIL's fake file system creation
  - LARIAT information assurance testbed
  - Biometric spoofing research

# Meta - Game Theory Formalizations

- Cat-and-mouse game of analysis system vs. malware
  - Strategy dependent on the “worthiness” of the adversary
  - Save advanced techniques for the most advanced opponent
- Stackelberg games
  - Allocation of analysis resources by analysis system with randomized strategy while malware deploys a purely deterministic evasion strategy

# Meta - Establishing Ground Truth

- *Unknown-unknowns*: researchers don’t know what they don’t know
- Human malware analysis is not scalable
- “Bootstrapping” corpora - use previously generated analysis reports as ground truth
    - Problematic: differences in execution environment and time may lead to spurious differences
- Collection in the wild
    - Challenging for *evasive* malware
    - Collection sources may reveal biases

# Meta - Challenges in Research Evaluation

- Evaluated works range from evaluating one lab-created malware sample to analyzing millions captured in the wild
- Impossible to empirically compare research, or reproduce results

- 2012: Rossow et al. - evaluated the “methodological rigor and prudence” of 36 papers involving malware experimentation from 2006-2011
  - We re-emphasize all of the author’s points and recommend researchers read their paper closely

# Meta - Suggestions for Improvement

- Establish ground truth
  - Verify analysis results for at least a portion of the malware with a human analyst
- Make multi-execution system similar
  - Minimize differences in environment causing spurious differences in execution
  - Discuss any unavoidable differences
- Be explicit about malware origins
  - Malware corpora may have inherent skews
    - VirusTotal - wild samples caught by defenders, or offensive actors doing testing
    - APTs - hard to catch

# Conclusion & Thank You

DEEPSEC

- Surveyed in paper: mobile and web analysis, case studies

- Continual evolution of offense and defense, will to continue into the future
- Cutting edge defenses may not be scalable
- Immense challenges to experimental evaluation and ground truth


- Friends who helped us edit: Rolf Rolles, James Kukucka, Aaron Sedlacek
- RPI support: Jeremy Blackthorne and Dr. Greg Hughes
- Program committee & our anonymous reviewers - particularly #4
- Dr. Sergey Bratus
- DeepSec / ROOTS



alexei@riverloopsecurity.com

yener@cs.rpi.edu