BotSwindler: Tamper Resistant Injection of Believable Decoys in VM-Based Hosts for Crimeware Detection

Presenting: Sal Stolfo
Columbia University

Collaborators

- Brian M. Bowen
- Pratap Prabhu
- Vasileios P. Kemerlis
- Stelios Sidiroglou
- Angelos D. Keromytis
- Salvatore J. Stolfo

Talk Outline

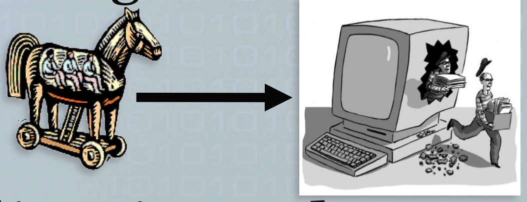
- Contributions
- Motivation
- Related Work
- Architecture
- Results of malware experiments
- Statistical and information theoretic analysis
- Conclusion

Contributions

- **BotSwindler architecture**
  - Tamper-resistant zero-day crimeware detection
- **VMSim language**
  - New language for expressing simulated user behavior
- **Virtual Machine Verification (VMV)**
  - Low overhead approach for verifying simulation state
- **Real malware detection results**
  - Financial bait credentials stolen and detected
- **Believability evaluation**
  - Statistical and information theoretic analysis
  - User study results

Motivation: Privileged Software

- External threat acquires insider privileges
  - Example: Spyware/Trojan Horse Programs



- Underground Economy trading in stolen digital credentials has spurred the growth of spyware
- Recent study focused of Zeus:
  - Over 3.6 million PC infections [Messmer09]
  - 55% bypassed up-to-date antivirus software [Trusteer09]

Related Work

- Borders *et al.:* malware attempting to blend in with normal user activity by manually injecting network requests [BZP06]
- Holz *et al.:* investigated keyloggers and dropzones, relied on executing maleware in CWSandbox and automation with AutoIt which runs in-host [HEF09]
- Egele *et al. and* Yin *et al.:* Taint analysis systems that work well, but with large overhead and in-host components [EKK07]
- Garfinkel *et al.:* VMI techniques, but none that rely on the VMM graphical framebuffer

Botswindler Approach: Deception

- Malware stealithly embedded and hard to detect via host behavior

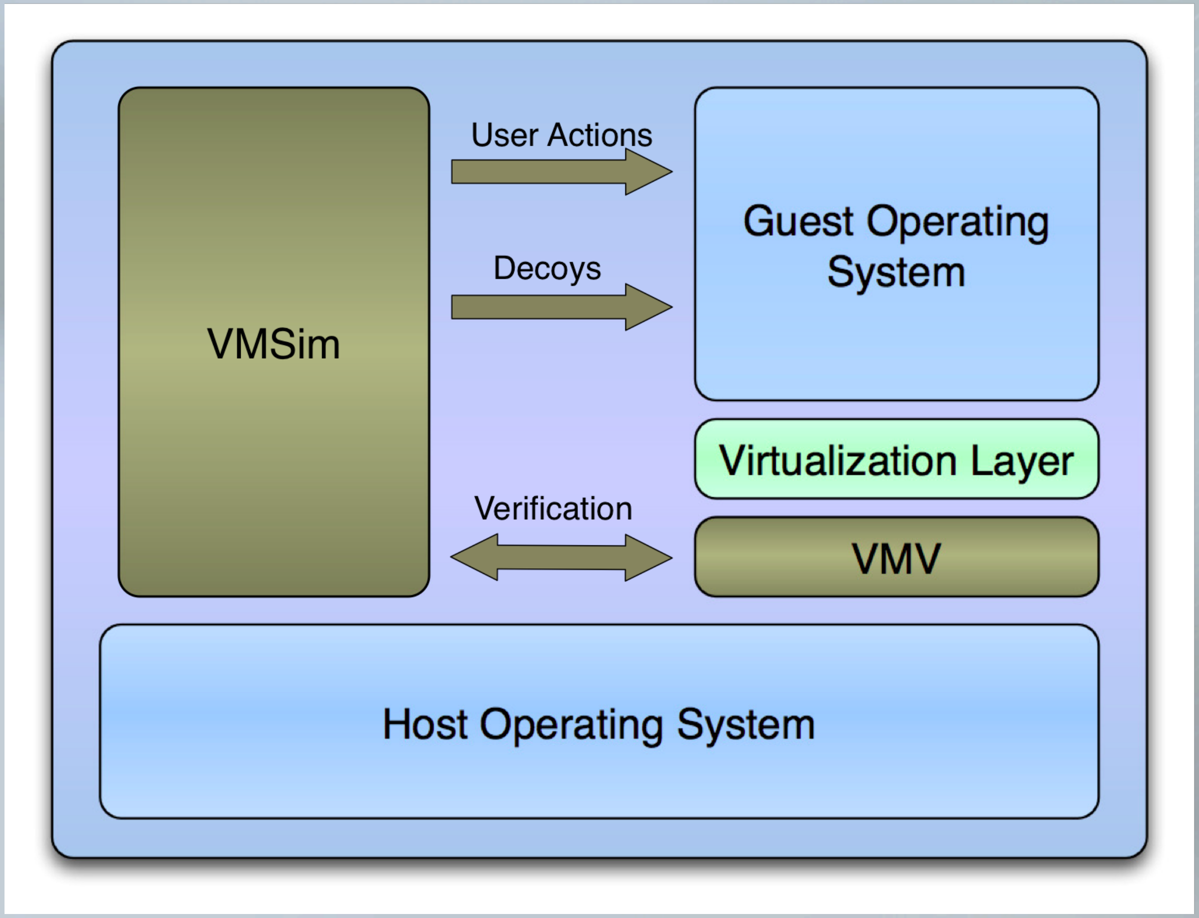
- Assume also malware inspects its environment to detect if it is being inspected in a VM/sandbox

- Deceive the malware into capturing “decoy’ credentials to reveal its presence when misusing those credentials!

Types of Decoys

- PayPal accounts
  - Created a set of decoy accounts tied to bogus identities
  - Created custom monitors to login into the account and poll last login time
  - If the last login time is not the same as previous polling time, an alert is generated
- Gmail accounts
  - Custom monitor checks to see if IP and time are inconsistent with monitors.
- Bank accounts from large financial institute
  - Daily reports from the bank give us the times and IPs of all accessed accounts

BotSwindler Components



VMSim Goals

- Goal 1: Simulator process remains undetected by the malware
  - Decouple the location of where the simulation process is executed and where its actions are received
  - Run simulator outside of a virtual machine and pass its actions to the guest host by utilizing the X-Window subsystem on the native host
- Goal 2: The actions of the simulator appear to be generated by a human
  - Simulation creation process entails recording, modifying, and replaying mouse and keyboard events captured from real users

VMSim

- Simulator runs on the native host producing human-like events without introducing technical artifacts that could be used to alert malware of the BotSwindler façade

- Formal Language:

$$
\begin{aligned}
&< ActionType > ::= < WinLogin >< ActionType >\\
&\quad | < CoverType >< ActionType > | < CarryType >< ActionType >\\
&\quad | < WinLogout > | < VerifyAction >< ActionType > | \epsilon\\
&< CoverAction > ::= < BrowserAction >< CoverAction >\\
&\quad | < WordAction >< CoverAction >\\
&\quad | < SysAction >< CoverAction >\\
&< BrowserAction > ::= < URLRequest >< BrowserAction >\\
&\quad | < OpenLink >< BrowserAction > | < Close >\\
&< WordAction > ::= < NewDoc >< WordAction >\\
&\quad | < EditDoc >< WordAction > | < Close >\\
&< SysAction > ::= < OpenWindow > | < MaxWindow >\\
&\quad | < MinWindow > | < CloseWindow >\\
&< VerifyAction > ::= Img1 \mid Img2 \mid ... \mid ImgN \mid Unknown\\
&< CarryAction > ::= < PayPalInject > | < GmailInject >\\
&\quad | < CCInject > | < UnivInject > | < BankInject >
\end{aligned}
$$

Virtual Machine Verification

- Primary challenge lies in generating human-like events in the face of variable host responses (network latency, OS issues, and changes to web content)
- Approach: decide whether the current VM state is in one of a predefined set of states.
- States are defined **manually** with graphical artifacts or pixel selections
- State monitoring is built into the VMM

Can Malware detect BotSwindler?

- Faulty simulations or virtual machine verification
- Statistical analysis of keystroke timing
- Variation in system operation/performance

Virtual Machine Verification Overhead

- Tables represent the amount of time in seconds to load web pages on a test machine
- Difficult to detect through performance differences

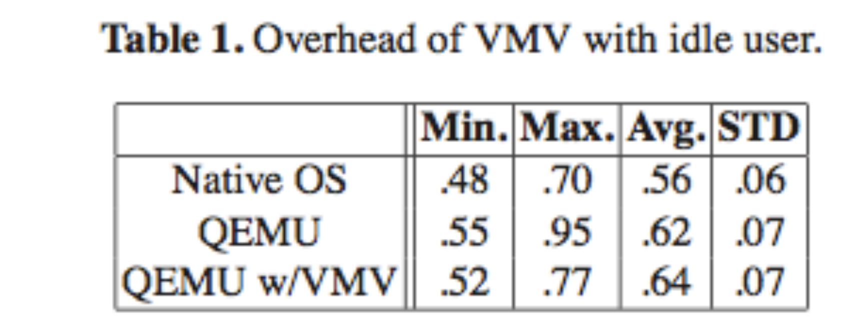


Table 1. Overhead of VMV with idle user.

	Min.	Max.	Avg.	STD
Native OS	.48	.70	.56	.06
QEMU	.55	.95	.62	.07
QEMU w/VMV	.52	.77	.64	.07

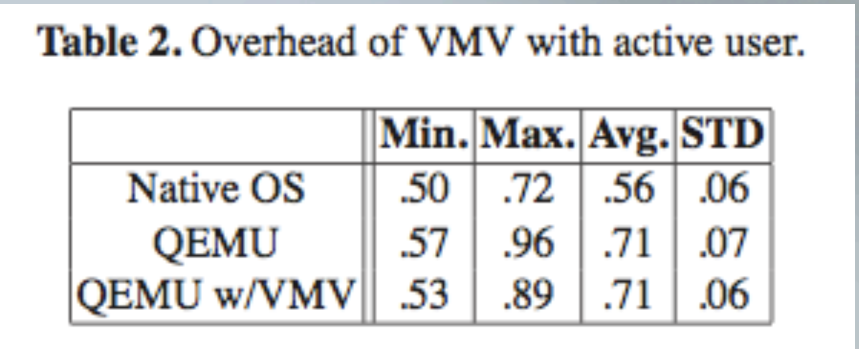


Table 2. Overhead of VMV with active user.

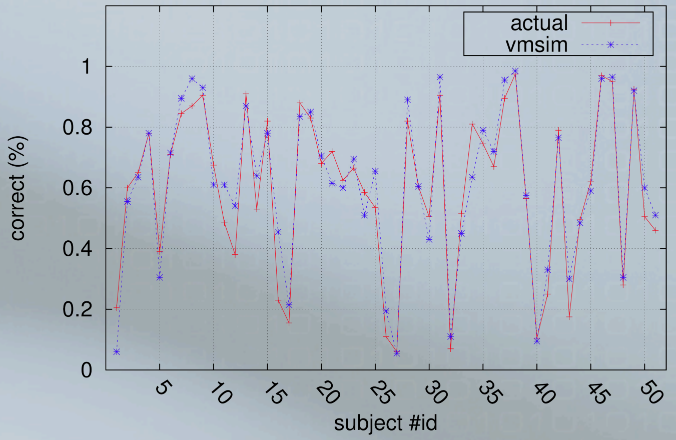
	Min.	Max.	Avg.	STD
Native OS	.50	.72	.56	.06
QEMU	.57	.96	.71	.07
QEMU w/VMV	.53	.89	.71	.06

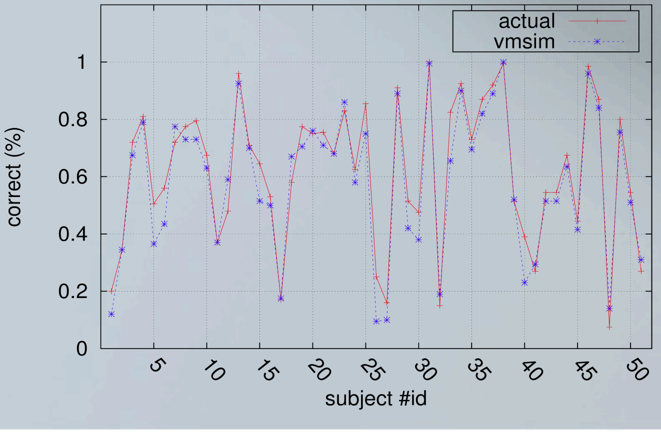
Statistical and Information Theoretic Analysis

- Goal: see if a ML algorithm might be able to classify keystrokes accurately into user generated or machine generated
- Relied on Killourhy and Maxion's benchmark data set for keystroke timing
  - created by having 51 subjects repeatedly type the same 10 character password, 50 times in 8 separate sessions, to create 400 samples for each user
- Our experiments:
  - Used 200 timing vectors for training classifiers
  - Used 200 other timing vectors for classifying against generated timing vectors

SVM and Naïve Bayes Classification Results

- Keystroke timing is indistinguishable



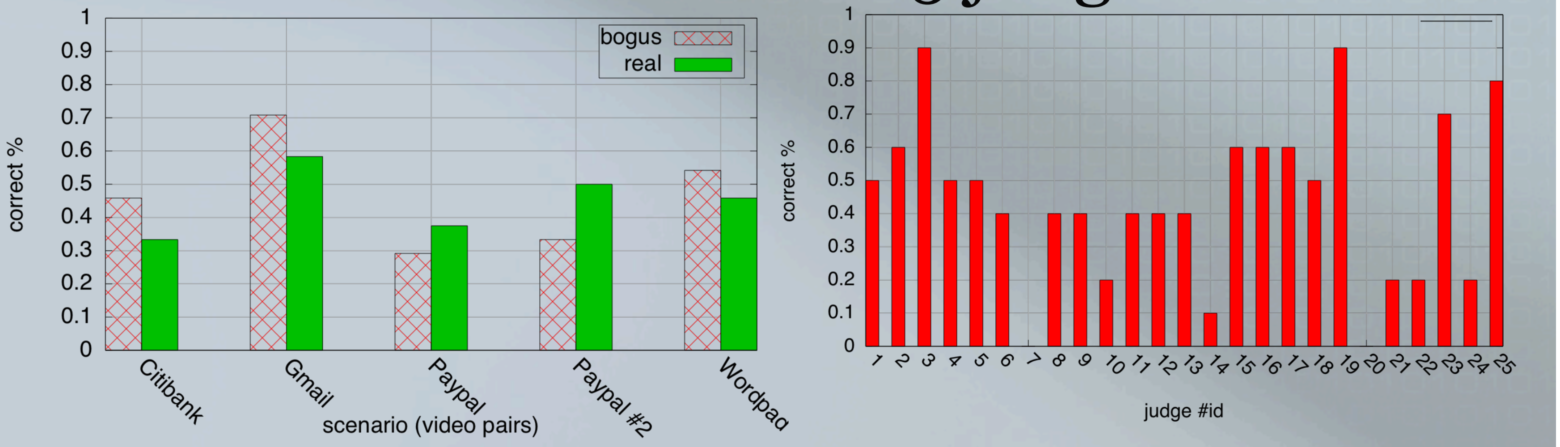


Decoy Turing Test

- Goal is to measure the believability of the simulations
- 25 human judges, consisting of security-minded PhDs, graduate-level students, and security professionals
- Tasked with observing a set of 10 videos that capture typical user actions performed on a host and make decision about each video: real or simulated

Decoy Turing Test Results

- The overall success rate was ~46%
- Optimal would be 50%
- Graphs show results for each of the 5 scenarios and each of the 25 judges



Experiments with malware

- Conducted experiments over 5 days using 116 Zeus variants from Swiss Security Blog
- 5 PayPal and 5 Gmail decoys
- Created phony PayPal site to give accounts enticing attributes (balance & verification)
- 20 minute simulation for each binary
- Results: 13 PayPal and 1 Gmail alert

Second Experiment with malware

- Relied on several bank accounts with balances exceeding $1,000
- In contrast to PayPal experiments, the bank site had authentic SSL certificates
- Ran the simulator for approximately 10 minutes on 59 new binaries
- We received 3 alerts from the collaborating financial institution in 5 days

Conclusion

- Decoy injection can be useful forensic tool for detecting crimeware that can be difficult to detect through traditional means.
- BotSwindler presents an instance of a system and approach that can be used to deal with information-level attacks, regardless of their origin

Conclusion – Future Work

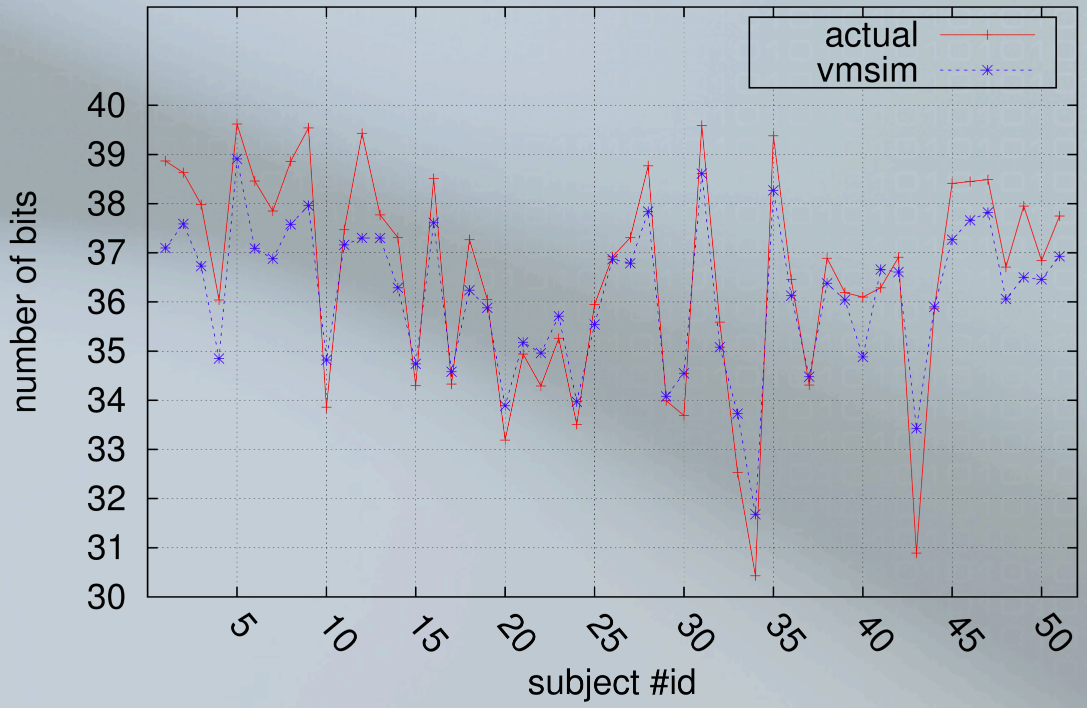
- Extending BotSwindler
  - Investigate methods for automating the porting of simulations from one host to another
  - Additional experiments with real bank accounts with real balances and tracking within the UE working collaboratively with an external organization

- Conduct experiments designed to demonstrate an expanded role of decoys for measuring organizational security and educating users

References

- [BZP06] Borders, K., Zhao, X., Prakash, A.: Siren: Catching evasive malware. In: Proc. of the IEEESymposium on Security and Privacy. pp. 78–85. Oakland, CA, USA (May 2006)
- [EKK07] Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D.: Dynamic spyware analysis. In: Proc. of the USENIX Annual Technical Conference. pp. 233–246. Santa Clara, CA, USA (June2007)
- [GR03] Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for in-trusion detection. In: Proc. of the 10th Annual Network and Distributed System Security Symposium (NDSS) (February 2003)
- [HEF09] Holz, T., Engelberth, M., Freiling, F.: Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones, Lecture Notes in Computer Science (LNCS), vol. 5789, pp. 1–18. Springer Berlin / Heidelberg (September 2009)
- [CMST06]Moore, A. P., Cappelli, D., Randall, T. F., "The Big Picture of Insider IT Sabotage Across U.S. Critical Infrastructures, CERT, Software Engineering Institute and CyLab at Carnegie Mellon University, 2007
- [Trusteer09] Measuring the in-the-wild effectiveness of antivirus against zeus. Technical report, Trusteer, September 2009.
- [Messmer09] E. Messmer. America's 10 most wanted botnets, July 2009.

Entropy Results


actual
vmsim
number of bits
subject #id