Run-time Classification of Malicious Processes Using System Call Analysis

Ray Canzanese
Dept. of Electrical and Computer Engineering
Drexel University
`rcanzanese@gmail.com`

Spiros Mancoridis
College of Computing and Informatics
Drexel University
`mancors@drexel.edu`

Moshe Kam
Newark College of Engineering
New Jersey Institute of Technology
`kam@njit.edu`

MALCON 2015
20-23 October
Fajardo, Puerto Rico

Acknowledgments

- The KEYSPOT Network
- People's Emergency Center
- Dornsife Center for Neighborhood Partnerships
- The City of Philadelphia Mayor's Commission on Literacy
- The City of Philadelphia Office of Innovation and Technology (OIT)
- The City of Philadelphia Department of Parks and Recreation (PPR)
- Secure and Trustworthy Cyberspace (SaTC) award from the National Science Foundation (NSF) – grant CNS-1228847
- The Isaac L. Auerbach endowed chair for Spiros Mancoridis

Setting

- Malware classification results are useful for generating
  - Mitigation procedures
  - Remediation procedures
  - Detection signatures
- Classification using sandbox environments is resource-intensive
- Malware authors generate variant floods to overwhelm analysts
- Analysts struggle to keep up with influx of new samples

We seek a classification system that

- Leverages endpoint monitoring
- Provides immediate classification results

Previous work

Related work

- Use static and dynamic analysis to classify malware samples[1] [2]
- Use sandbox environments for off-line analysis
- Leverage various datasets
  - Program structure, resources
  - File, registry, network, system call activity

Our approach

- Uses dynamic analysis (system call sequences)
- Focuses on on-line analysis
  - Uses endpoint monitoring for feature extraction
  - Does not require specialized sandbox environments
  - Can provide immediate classification results

[1] Neugschwandtner, “Forecast: skimming off the malware cream,” 2011.

[2] Anderson, “Improving malware classification: bridging the static/dynamic gap,” 2012.

Hypothesis

Classify malware by

- Monitoring system call activity on endpoints
- Extracting a concise feature representation of the traces
- Comparing observed patterns to those of known malware

Advantages

- Monitoring and extraction are low-overhead
- Classification results can be obtained at run-time
- Can be easily paired with static analysis techniques
- Availablility of results facilitates analysis

Impact and broader contributions

- Feature extraction and classification algorithm comparison
  - 3 feature extraction strategies
  - 6 machine learning algorithms
  - Analysis of trace length and n-gram length
- Ground truth labeling system comparison
  - 27 naming schemes derived from AV labels
  - Category and family naming schemes
- Design of a run-time classification system
  - Algorithms and parameters based on experimental evaluation
  - Evaluated against $76{,}000$ distinct malware samples
  - Enables more rapid response to newly disovered malware treats

System call analysis

Inferring a process's function from its system call trace[3]

System call

- Mechanism for requesting operating system (OS) services

System call categories

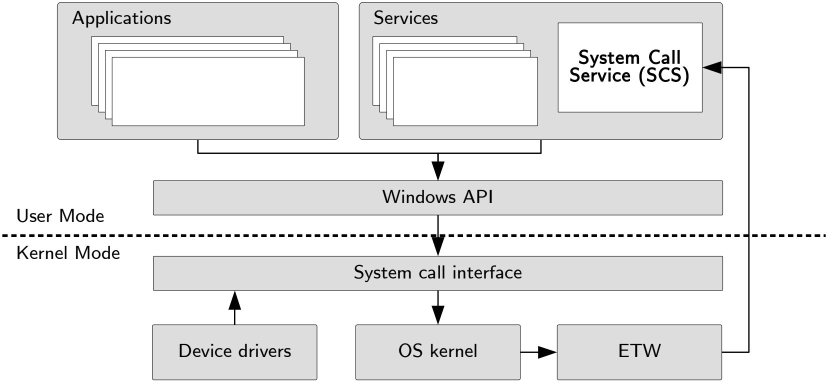
- Atoms (strings)
- Boot configuration
- Debugging
- Device driver control
- Environment settings
- Error handling
- Files and general input/output
- Jobs
- Local procedure calls (LPC)
- Memory management
- Miscellaneous
- Object management
- Plug and play
- Power management
- Processes and threads
- Processor information
- Registry access
- Security functions
- Synchronization
- Timers

[3] Forrest, "A sense of self for UNIX processes," 1996.

System Call Service (SCS)

Data collection host-agent[4]

- Designed for Windows 7, 8, Server 2008, and Server 2012 (32 and 64 bit)
- Collects process-level system call traces from all processes



[4] SCS source code available: `https://github.com/rcanzanese/SystemCallService`

Information retrieval

Bag-of-system-call-n-grams representation[5]

Raw system call trace:

```
NtQueryPerformanceCounter
NtProtectVirtualMemory
NtProtectVirtualMemory
NtQueryInformationProcess
NtProtectVirtualMemory
NtQueryInformationProcess
```

Representation:

system call 2-gram bag	count
NtQueryPerformanceCounter, NtProtectVirtualMemory	1
NtProtectVirtualMemory, NtProtectVirtualMemory	1
NtProtectVirtualMemory, NtQueryInformationProcess	2
NtQueryInformationProcess, NtProtectVirtualMemory	1

[5] Kang, "Learning classifiers for misuse and anomaly detection using a bag of system calls representation," 2005.

Feature scaling

- Term frequency – inverse document frequency (TF-IDF) transformation[6]
  - De-emphasize commonly occurring n-grams
- Singular value decomposition (SVD)[7]
  - Reduce the dimensionality of the data
  - Eliminate redundancy
- Linear discriminant analysis (LDA)[8]
  - Reduce the dimensionality of the data
  - Separate instances of differing classes

6 Liao, "Using text categorization techniques for intrusion detection," 2002.
7 Manning, *Introduction to Information Retrieval*, 2008.
8 Bishop, *Pattern Recognition and Machine Learning*, 2006.

Classification

- Multi-class logistic regression (LR)[9]
  - One-versus-all approach using stochastic gradient descent (SGD)
  - Assume linearly separable classes
- Naive Bayes[10]
  - Estimate priors from data
  - Assume conditional independence
- Random Forests[11]
  - Realize non-linear decision surfaces
  - High training complexity
- Nearest neighbor[12]
  - Realize non-linear decision surfaces
  - High model & classification complexity
- Nearest centroid[13]
  - Assume equal variance and class convexity

[9] Genkin, "Large-scale Bayesian logistic regression for text categorization," 2007.

[10] VanTrees, *Detection, Estimation, and Modulation Theory*, 2001.

[11] Breiman, "Random forests," 2001.

[12] Bishop, *Pattern Recognition and Machine Learning*, 2006.

[13] Han, "Centroid-based document classification: analysis and experimental results," 2000.

Evaluation

$FN_{\mathcal{C}_k}$	false negatives
$TP_{\mathcal{C}_k}$	true positives
$FP_{\mathcal{C}_k}$	false positives

$$\text{Precision}_{\mathcal{C}_k} = \frac{TP_{\mathcal{C}_k}}{TP_{\mathcal{C}_k} + FP_{\mathcal{C}_k}}$$

$$\text{Recall}_{\mathcal{C}_k} = \frac{TP_{\mathcal{C}_k}}{TP_{\mathcal{C}_k} + FN_{\mathcal{C}_k}}$$

$$F_{1,\mathcal{C}_k} = 2 \cdot \frac{\text{Precision}_{\mathcal{C}_k} \cdot \text{Recall}_{\mathcal{C}_k}}{\text{Precision}_{\mathcal{C}_k} + \text{Recall}_{\mathcal{C}_k}}$$

Ground truth label comparison

vendor	type	classes	F_1
AntiVir	category	17	0.79
Microsoft	category	20	0.75
DrWeb	category	12	0.75
Microsoft	family	315	0.71
Vipre	category	47	0.71
ESETNOD32	family	301	0.68
Panda	category	19	0.68
Avast	category	12	0.66
K7AntiVirus	category	16	0.65
DrWeb	family	241	0.59
...	...	...	...
McAfee	family	125	0.53
Panda	family	111	0.53
Ikarus	family	442	0.5
Kaspersky	family	290	0.49
FSecure	family	175	0.48
Emsisoft	category	73	0.48
Avast	family	220	0.47
TrendMicro	family	227	0.46
GData	family	261	0.43
Emsisoft	family	293	0.43

Classifier and feature extraction strategy comparison

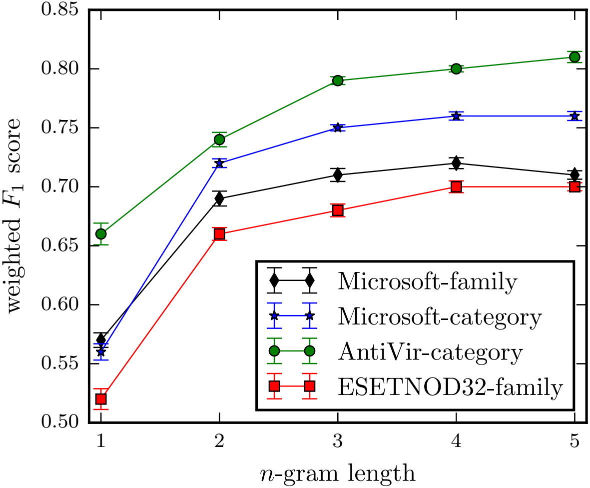
detector	feature extraction	F_1
LR	TF-IDF	0.70
nearest neighbor	TF-IDF, SVD	0.67
nearest neighbor	TF-IDF, SVD, LDA	0.67
random forests	TF-IDF, SVD	0.67
random forests	TF-IDF, SVD, LDA	0.67
LR	TF-IDF, SVD, LDA	0.56
LR	TF-IDF, SVD	0.53
Gaussian naïve Bayes	TF-IDF, SVD, LDA	0.50
nearest centroid	TF-IDF, SVD, LDA	0.42
Gaussian naïve Bayes	TF-IDF, SVD	0.39
multinomial naïve Bayes	TF-IDF	0.33
nearest centroid	TF-IDF, SVD	0.19

Other advantages of LR:

- Low classification complexity
- Model can easily be updated when new training instances are added

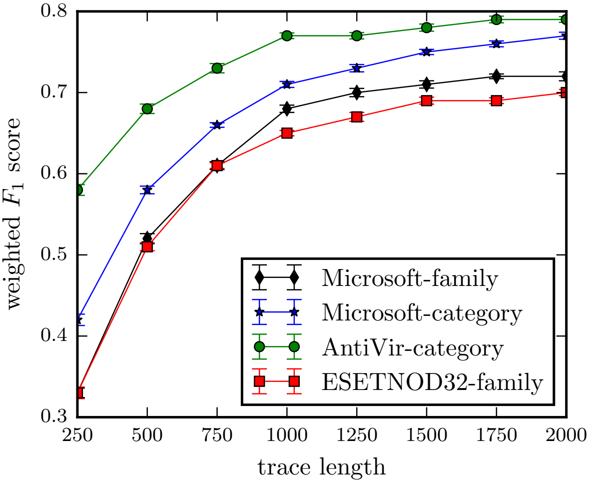
Classification accuracy vs. n-gram length

Fixed trace length, $l = 1500$

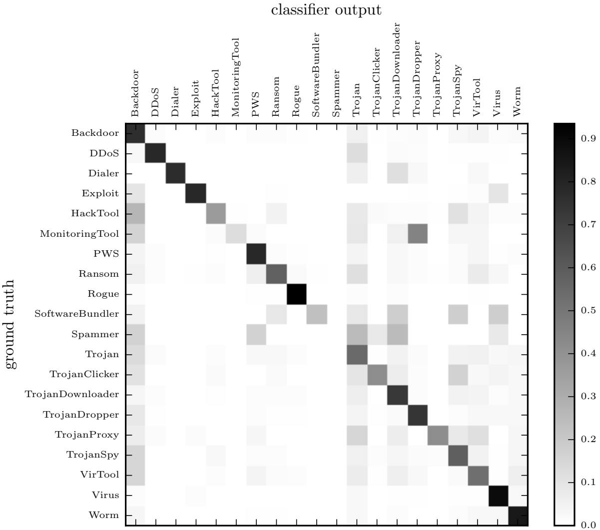


Classification accuracy vs. trace length

Fixed n-gram length, $n = 3$



Categorical confusion matrix



Malware family reults

Microsoft MMPC labels

Highest classification accuracy

Narrowly defined families

- Trojan.Mydoom
- Trojan.Recal
- Trojan.Jeefo
- Worm.Klez
- Virus.Elkern

Lowest classification accuracy

Broadly defined families

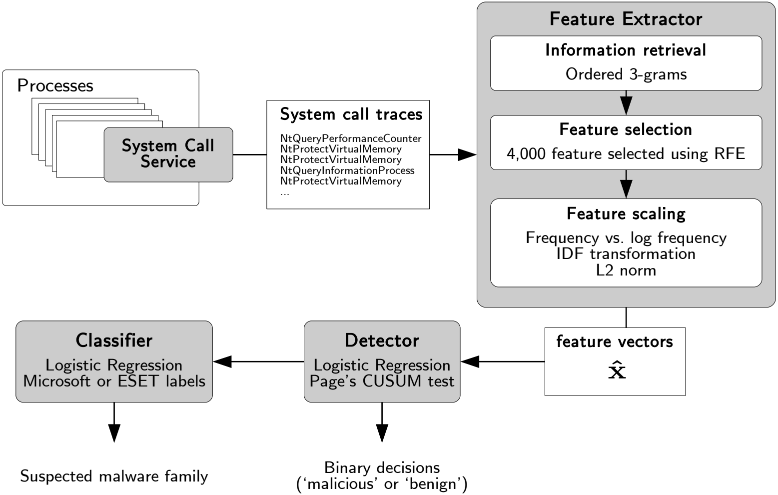
- Trojan.Meredrop
- Trojan.Gandlo!gmb
- Trojan.Ircbrute!gmb
- Trojan.Sisron!gmb
- VirTool.Vtub

MALCON

IEEE computer society

System block diagram

Shows classifier integrated with a system call-based detection system



Observations

Classification accuracy is dependent on:

- Ground truth labeling system
  - Family-level labels provide most meaningful results
  - MMPC and ESET labels provide highest accuracy
- Feature extraction strategy
  - Trace lengths of at least 1500 system calls
  - n-gram lengths of at least 3
  - TF-IDF feature scaling
- Classification algorithm
  - Multi-class logistic regression

Summary and conclusions

Objective

Classify malware at run-time in production environments based on easily observable characteristics

- Feature extraction and classification comparison
  - Compared multiple feature scaling techniques and model parameters
  - Compared multiple classifiers
- Evaluated the effects of ground truth labeling strategies
  - Derived labels from AV naming systems
  - Evaluated classifiers using category and family labels
- Presented the design of a run-time classification
  - Evaluated against $76{,}000$ malware samples run in production environments
  - Established through experimental evaluation

Remaining questions

- How well can classifier differentiate among classes of benign behavior?
- How easily can malware authors manipulate classification results?
- How do unsupervised approaches (clustering) compare?
- Are there more meaningful classes to use (remediation strategies)?
- How to improve results for poorly performing classes?
- How can this approach be paired with other approaches (static)?

Run-time Classification of Malicious Processes Using System Call Analysis

Ray Canzanese
Dept. of Electrical and Computer Engineering
Drexel University
rcanzanese@gmail.com

Spiros Mancoridis
College of Computing and Informatics
Drexel University
mancors@drexel.edu

Moshe Kam
Newark College of Engineering
New Jersey Institute of Technology
kam@njit.edu

MALCON 2015
20-23 October
Fajardo, Puerto Rico

MALCON

IEEE computer society