

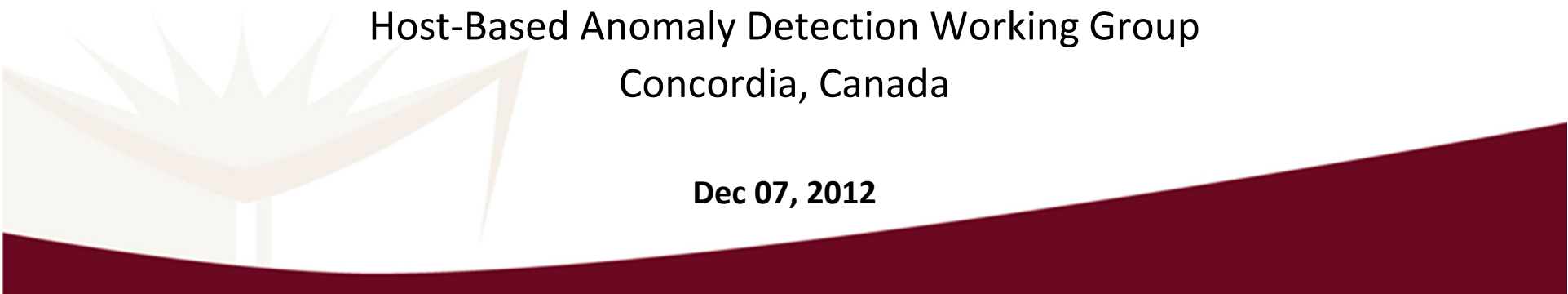
Online surveillance of critical computer systems through advanced host-based detection

Harmonized Anomaly Detection Techniques Thread

Shariyar Murtaza, Shayan Eskandari,
Afroza Sultana , Wahab Hamou-Lhadj

Software Behaviour Analysis Research Lab
Host-Based Anomaly Detection Working Group
Concordia, Canada

Dec 07, 2012



Contents

- Progress on kernel rootkit detection
- Progress on system call based anomaly detection for applications



What is a Rootkit?

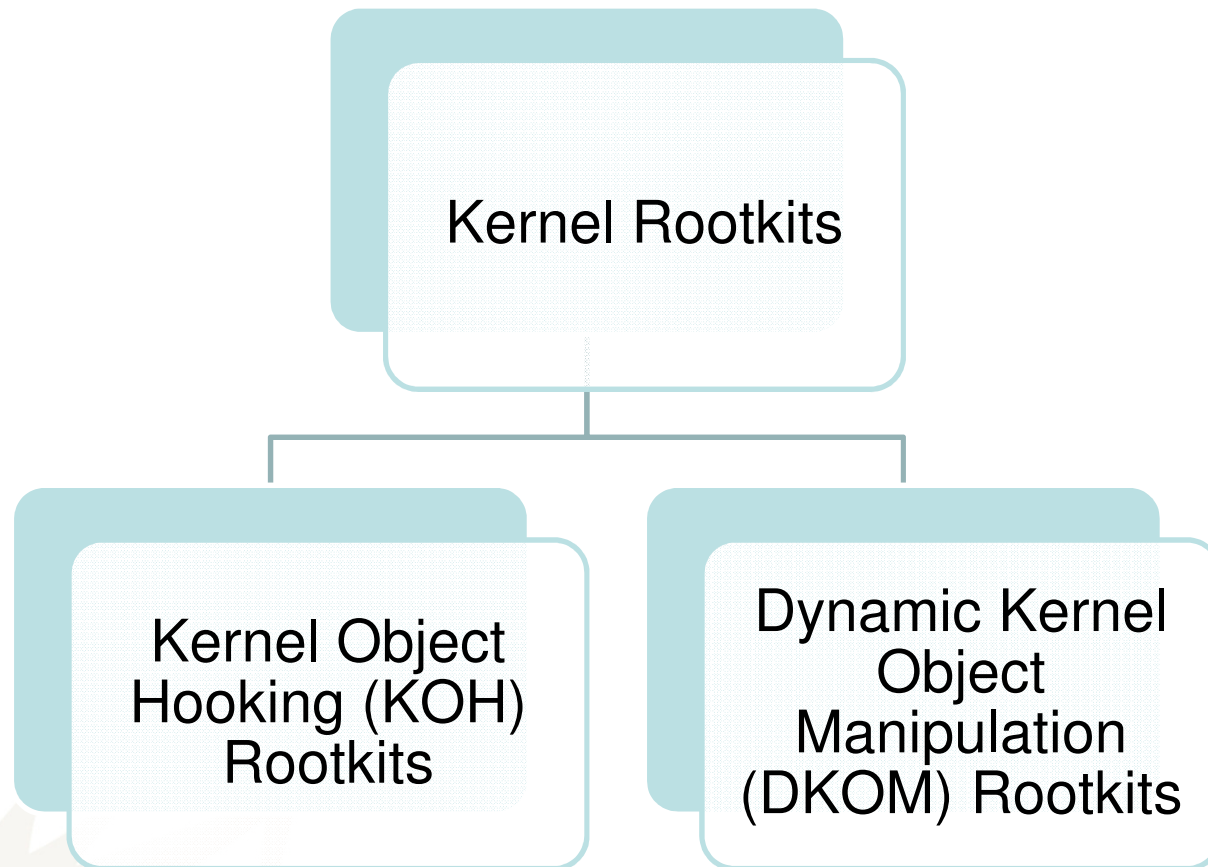
- Rootkit is a malware having several functionalities:
 - Stealth processing
 - Covert communication from system administrators.
 - Keystroke logging
 - Packet sniffing
 - Backdoor shell access
 - Remote attacking on networks

Severity of Rootkits

- Data-theft accounts for 80% of all cyber crimes
- Some recent examples of rootkits activities are:
 - Bank frauds
 - Disabling antivirus software
 - Making a system a bot
 - Stealing information

Source: [DARPA Technical Report, 2007], [McAfee Whitepaper, 2009]

Types of Kernel Rootkits



Main Types of Kernel Rootkits

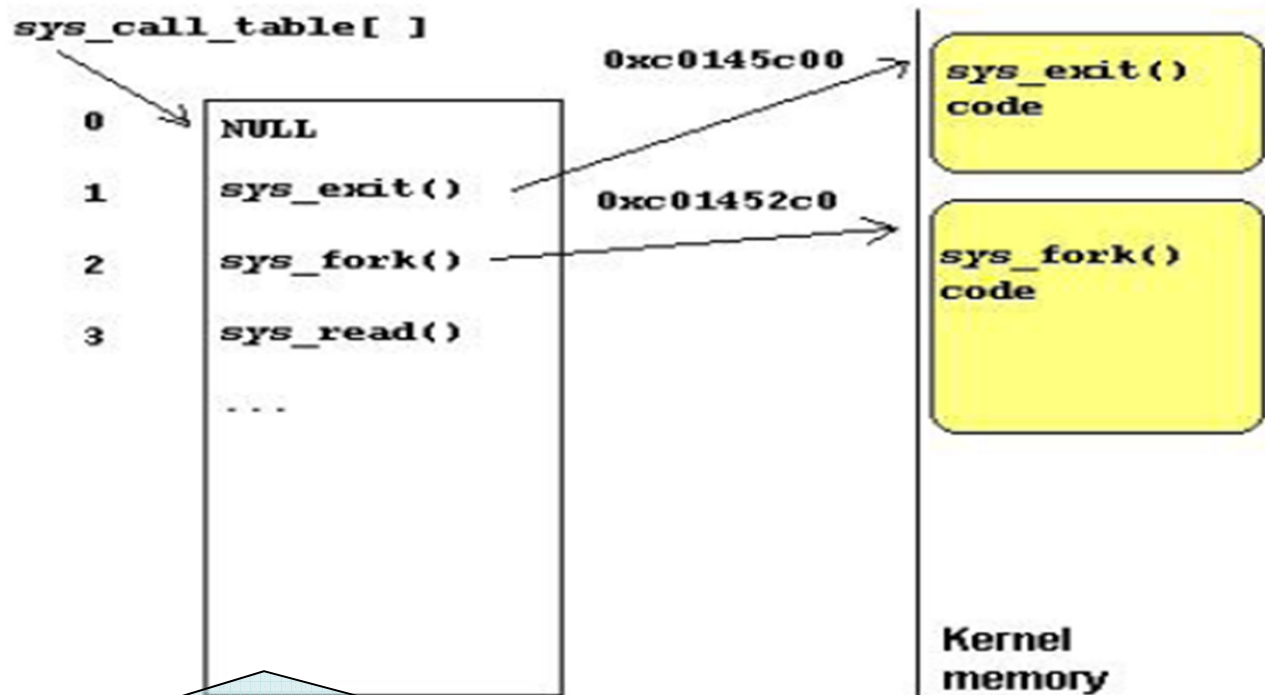
KOH

- Modify control data structures
- A handler function registers its address with kernel data structures; e.g., syscall hooking

DKOM

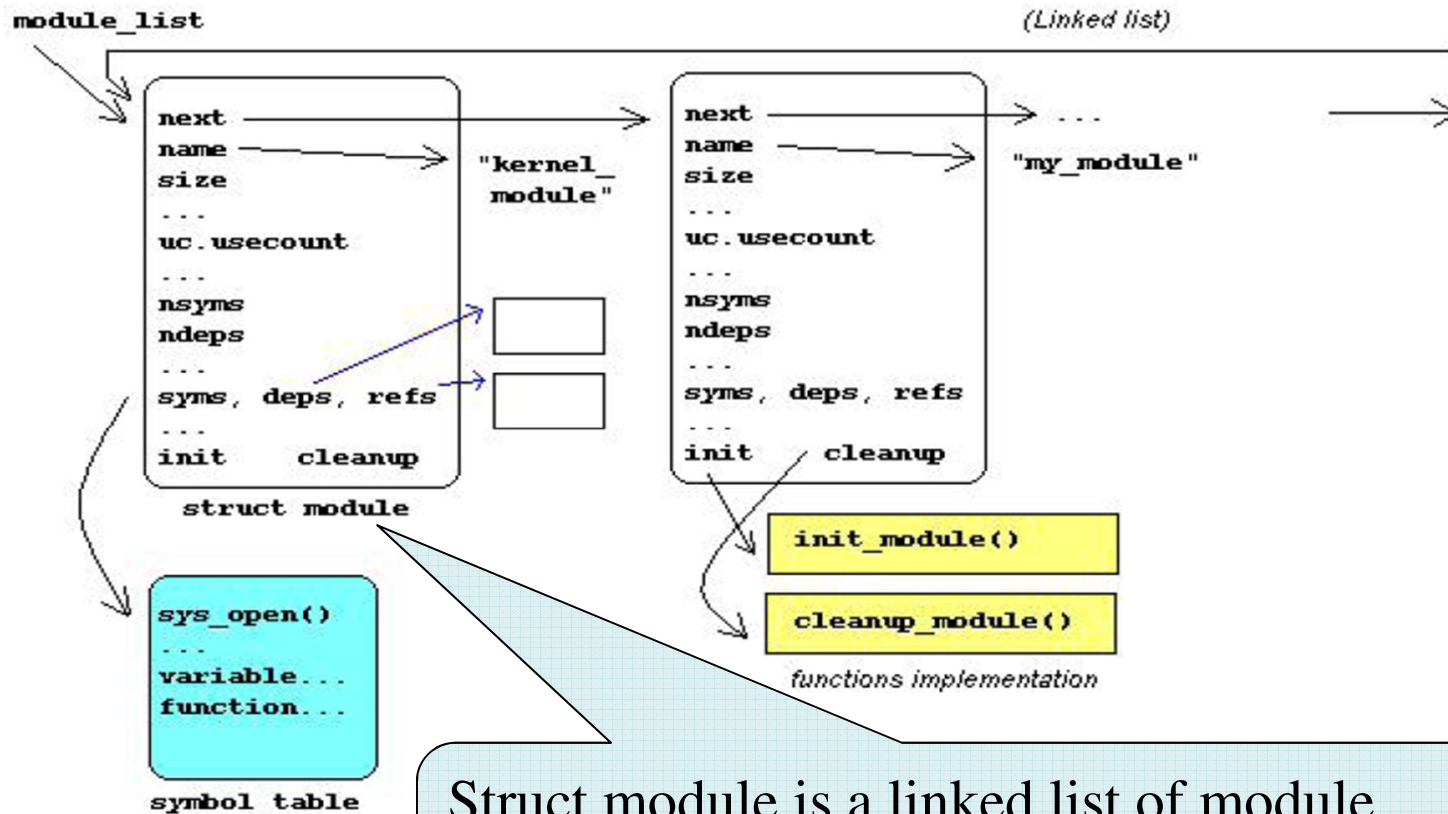
- Modify non-control data structures
- Manipulate internal record-keeping data within main memory; e.g., the list of running processes

KOH Rootkit



`sys_call_table` structure contains a set of pointers to functions implementing various system calls. A system call can be overridden by changing pointers.

DKOM Rootkit



Struct module is a linked list of module objects. An LKM can be hidden by removing its entry from this struct.

Common Methods Used by Rootkits

Loadable Kernel Module (LKM) can replace underlying system calls with their own version; e.g., Knark, Adore-ng

Directly patch the kernel's virtual memory (/dev/kmem) or physical memory (/dev/mem); e.g., SuckIT, Super User Control Kit

Directly patch the kernel's image on hard disk (/boot/vmlinuz); e.g., Kpatch

Using virtual machine to run a fraudulent system; e.g., BluePill

Using libc crashes to execute kernel instructions through stack for malicious purpose; e.g., return-to-libc rootkits

Rootkit Detection and Prevention Techniques

Host based techniques

Virtualization based techniques

External observer based techniques



Host Based Techniques

Techniques	Description
Kruegel et al. [2004]	Detect malicious LKMs using static analysis of LKM binaries
Kroah-Hartman [2004]	Load only RSA encrypted signed modules into memory
Secure boot [Parno et al., 2010;Jaeger et al.,2011].	Load a component if the hash is equal to a known-good value
Jestin et al. [2011a]	Cluster memory addresses to detect high memory addresses related to malicious system calls
AppArmor [Bauer, 2006] and SELinux [Smalley et al., 2002]	Limit access to the kernel by using policies
Strider Ghostbuster [Beck et al., 2005]	Identify hidden files and processes using normal views

Host Based Techniques: Tools Scanning Known Places

- Kstat—/dev/kmem vs. system.map
- Kern check—*system.map* vs. system call table
- Chkrootkit—logs and configs
- Rootkithunter—files, ports, processes
- Rkscan—Adore, Knark
- Knarkfinder—hidden processes
- Tripwire, Samhain and AIDE—checksum based integrity
- Sleuth Kit—File system forensics tool

Virtualization Based Techniques

Techniques	Description
[Garfinkel & Rosenblum, 2003]	Enforce HIDS policies from VMM, such as signature scan of memory, comparing commands, text comparison, etc.
[Petroni et al. 2007]	Use cryptographic hashes of code and the graph of function pointers to detect control flow (KOH) anomalies
[Wang et al. 2009]	Make a copy of hooks (pointers) to a write protected location, verify accesses and prevent KOH rootkits
[Seshadri et al., 2007] and [Riley et al., 2008]	Prevent kernel code from unauthorized modification and execution—targets KOH rootkits.
[Baliga et al. 2008]	Prevent KOH rootkits by using the policies based on process and file relationships
[Rhee et al. 2009]	Use policies for key data structure (e.g., modification through known functions) to detect DKOM rootkits
[Jiang et al. 2007]	A technique to run anti-malware programs from outside of an OS on a VM; e.g., antivirus

External Observer Based Techniques

- Copilot [Petroni et al., 2004], a PCI-card monitor, compares kernel text, LKM text and function pointers to detect KOH rootkits
- Gibraltar [Baliga et al., 2011] detect KOH and DKOM rootkits by using data structure invariants

Purpose	Invariant	Description
Detect hidden process	$\text{run-list} \subset \text{all-tasks}$	run_list is a process list used by scheduler and all_task by others
Don't let firewall disable	$\text{nf_hooks}[2][1].\text{next.hook} == 0xc03295b0$	To avoid redirection actual address is identified

Classification of Anti-Rootkit Techniques

	Type	KOH	DKOM
Static analysis of binary images of LKM [Kreugel et al., 2004]	HB	Yes	No
Rootkit hunter [RootkitHunter]	HB	Yes	Yes
State based control flow integrity monitor [Petroni et al. , 2007]	VM	Yes	No
HookSafe [Wang et al., 2009]	VM	Yes	No
KernelGuard [Rhee et al. 2009]	VM	No	Yes
Gibraltar [Baliga et al., 2011]	EM	Yes	Yes
NICKLE [Riley et al, 2008]	VM	Yes	No

HB= Host Based; VM= Virtual Machines; and EM= External Monitor

Lessons Learned

1

Control flow integrity results in few or no false positives

2

DKOM rootkits can be detected by monitoring data structures and legitimate modifier functions

3

Return oriented rootkits can be detected if the instructions they push on stacks change the normal flow of execution

Challenging Solution

Monitor the call graph and data structure modifications

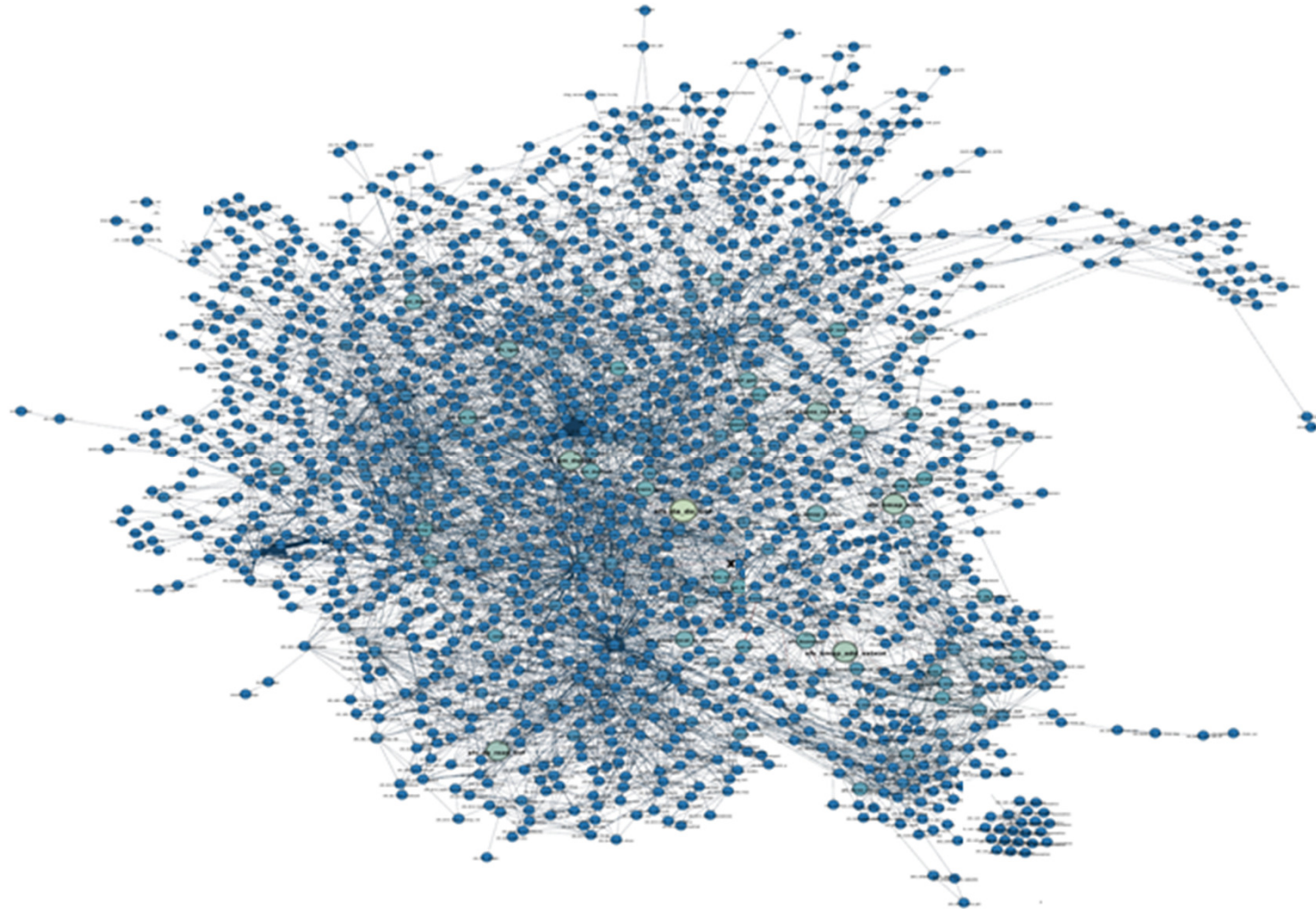
Monitoring the entire control graph of the kernel will cause a very high overhead



Statistics for Kernel 2.6.32.44

Description	Value
Total functions in source code (approx.)	232312
Total functions excluding documentation, scripts and drivers (approx.)	107094
Graph size (total edges)	394212
Callers	81919
Callees	75426

A Routine Call Graph of the “fs” Subsystem



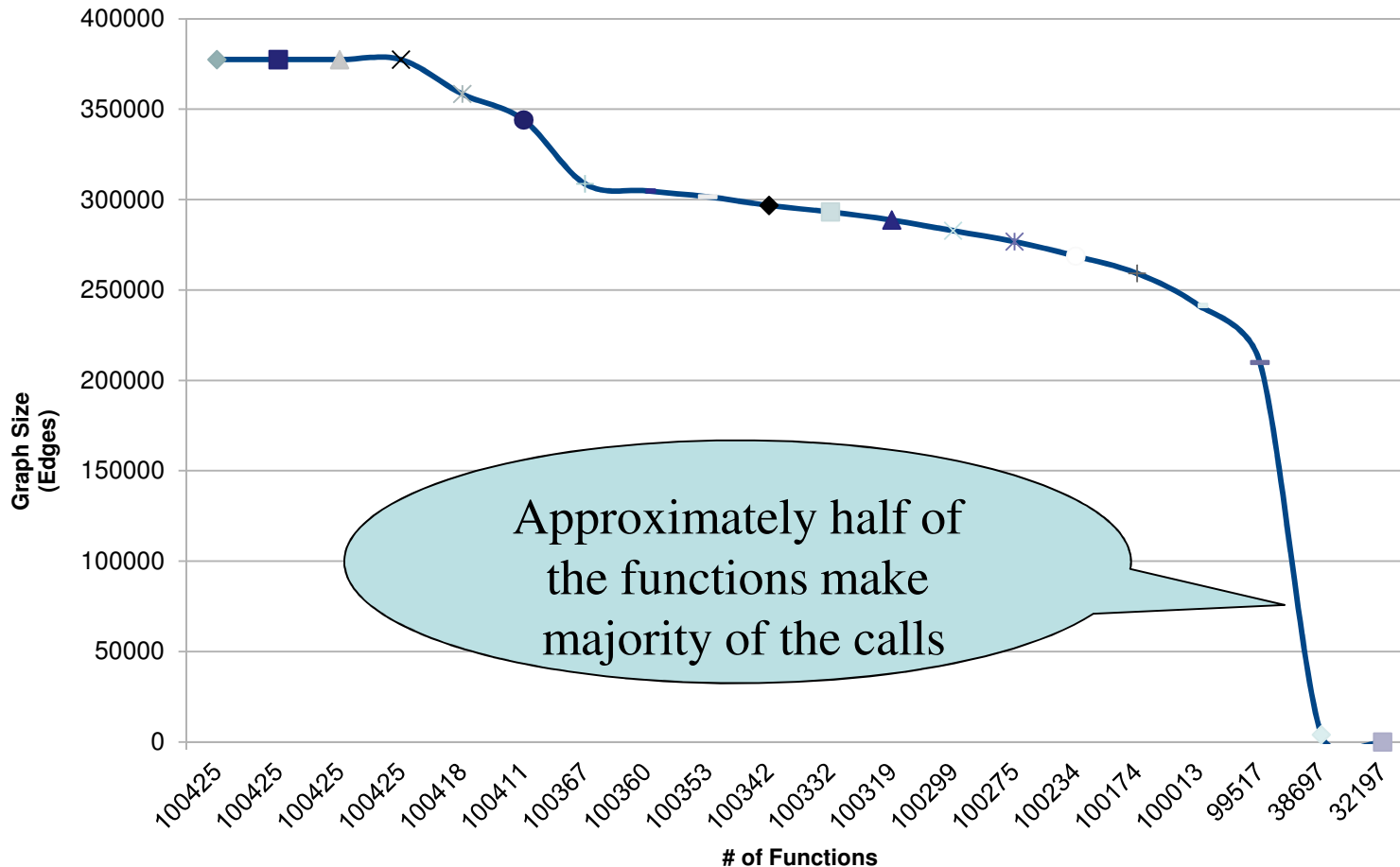
Proposed Solution: Traffic-Based Approach

- Based on Hamou-Lhadj's empirical studies on identifying important components in large systems:
 - High traffic vs. Low traffic components
 - Only a small number of functions make the largest number of calls
 - Priority monitoring should focus on only these functions
 - Secondary monitoring: Other functions can be added as needed

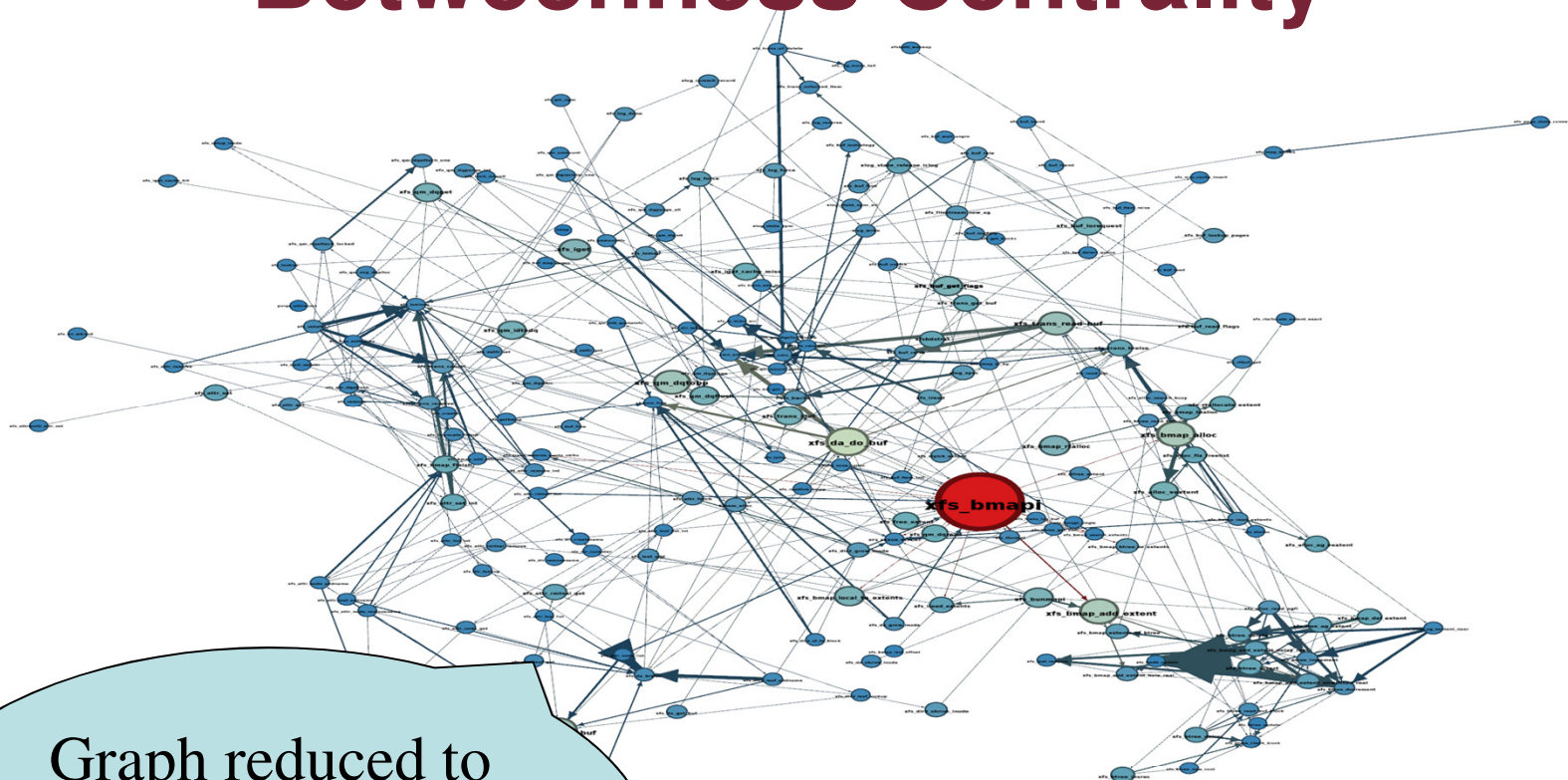
Identifying Important Functions

- Look for functions that generate the largest number of calls – Hamou-Lhadj's work on identifying utilities in large systems
- Network-based techniques: Betweenness centrality analysis
- Look for the functions of most targeted components
- Knowledge oriented techniques: Study sensitive paths where attacks have the highest information gain

Reducing Overhead Using Calling Relationship



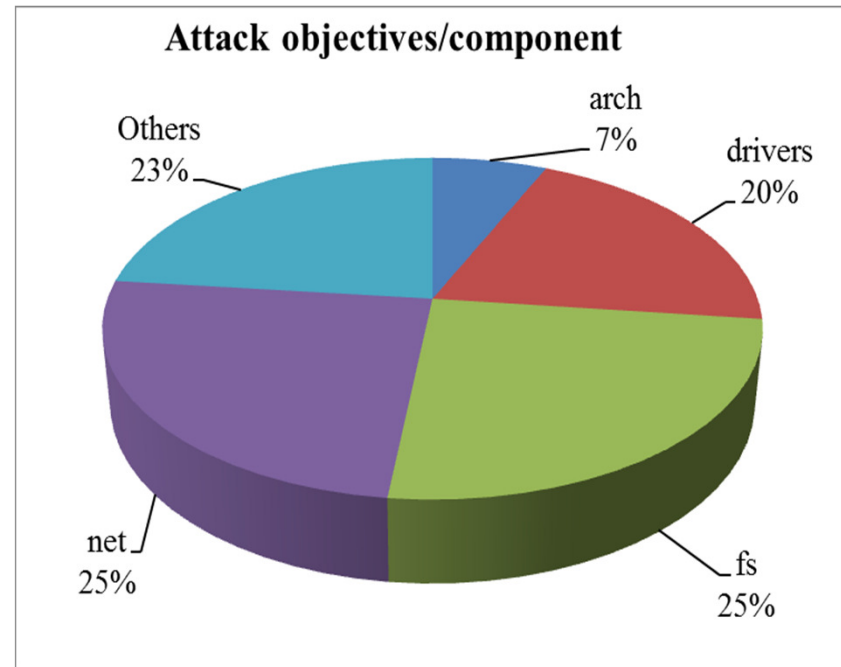
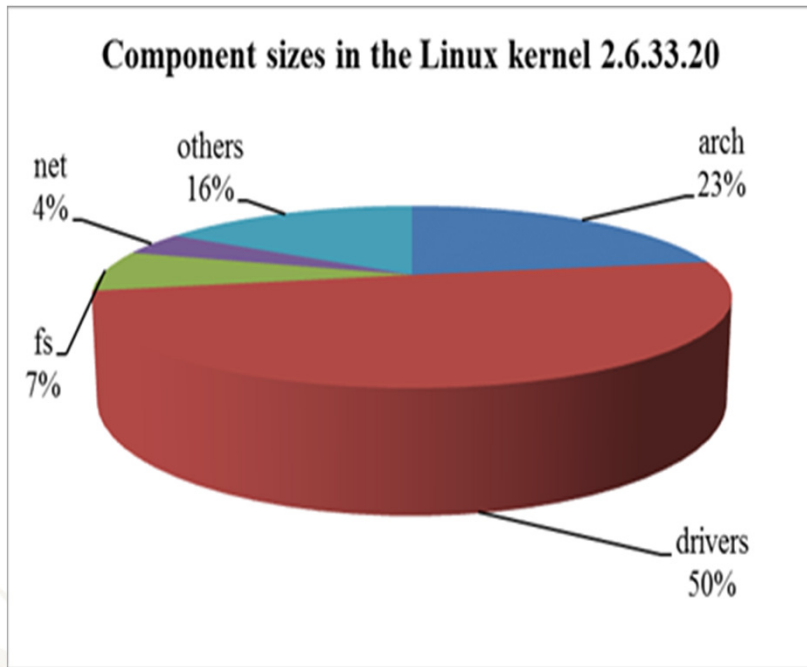
Reducing Overhead Using Betweenness Centrality



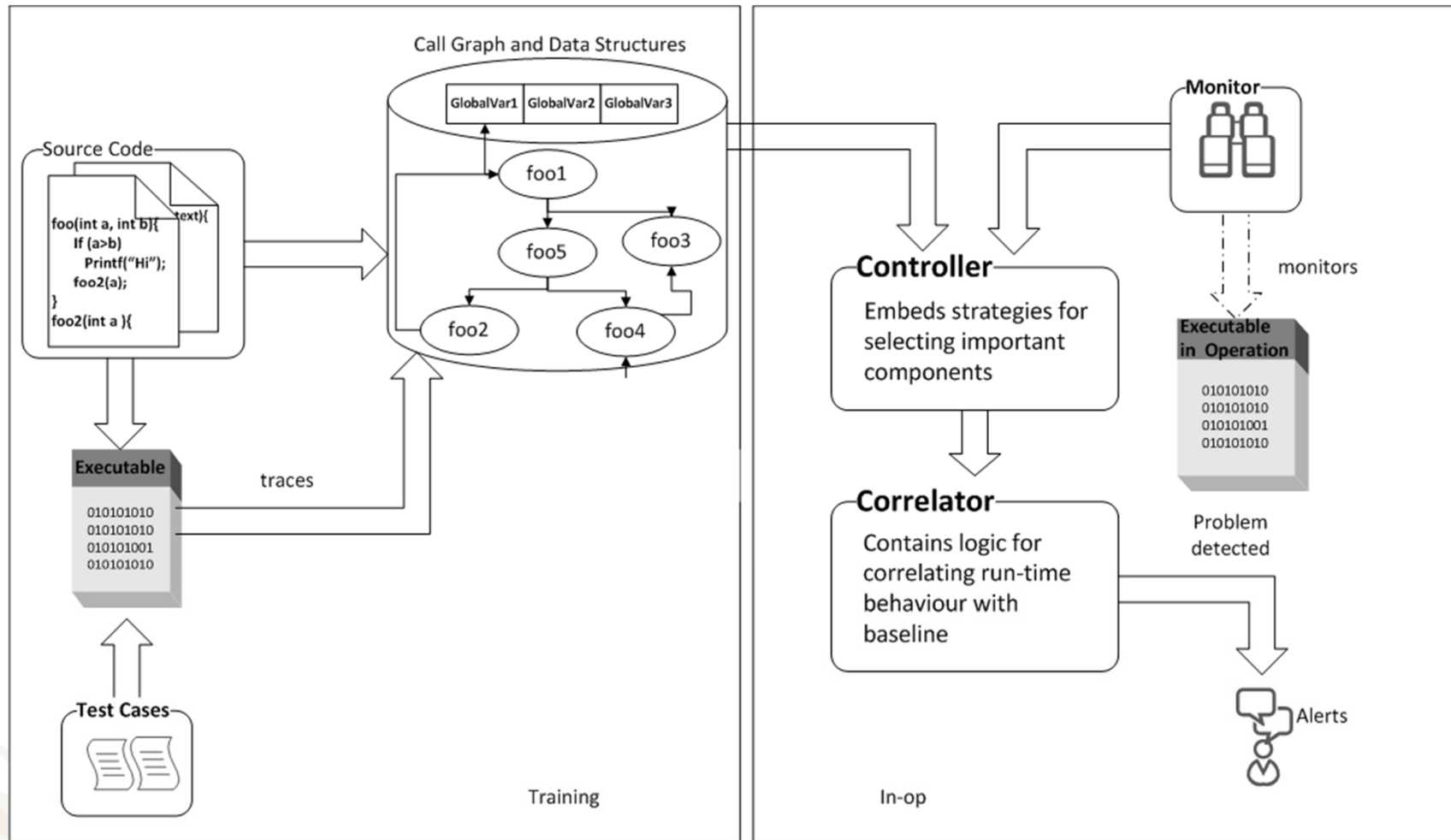
Graph reduced to
just the nodes with
high Betweenness
Centrality

Reducing Overhead Using Key Components

Distribution of 2009-2011 Linux vulnerabilities across components.



Vertex Framework

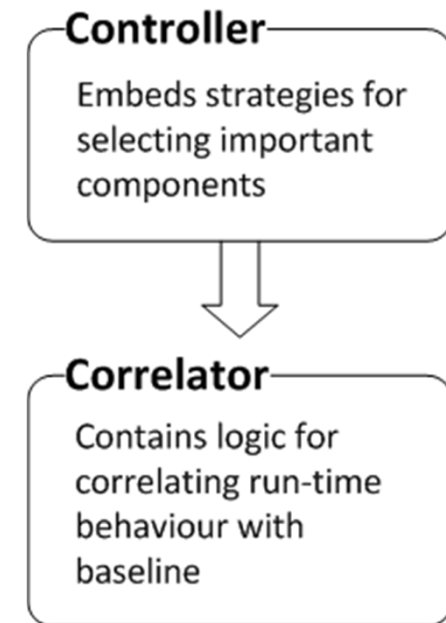


Preliminary Results

- Generated static and dynamic call graph for kernel
- Tested on a home grown rootkit and KBeast rootkit
 - Syscall hooking, key logging, process hiding, directory hiding, port hiding and backdoor shell access
- Both rootkits were detected immediately—unknown functions hooked to pointers
- Some false positives due to missing edges due to function pointers

Ongoing Tasks

- Studying the strategies for reducing graph size
- Developing correlation algorithms
- Developing a prototype tool
- Conducting experiments with real rootkits



Input For the Tracing Team

- Function calls and data structure tracing
- Notifications for global data structure modifications
- Develop a kernel stack monitor
- Develop a security mechanism against tampering of the monitoring system
- Easy to use interface to turn on and off probes

System Call Based Models For Applications



System Call Sequence Modeling

- Models an application's normal behavior from system calls sequences:
 - Sliding Windows [Forrest 1997, Warrender 1999]
 - Rule Based [Tandon 2003, Petrussenko 2010]
 - Neural Networks [Ghosh 1999]
 - Hidden Markov Models (HMM) [Hoang 2003, Hu 2009]
 - Finite State Automata (FSA) [Wagner 2001, Sekar 2001]
 - Variable length N-gram [Wespi 1999, Jiang 2002]
 - Statistical Techniques [Ye 2001, Burgess 2002]
 - Call Stack Techniques [Feng 2003]
 - Bag of System Call Technique [Kang 2005]

Attack sequences are **very similar to the normal system call** sequences—**false positives.**

Imitate legitimate system call sequences that execute malicious code—**which calls are they?**

Limitations of System Call Sequence Models

Replace the **system call arguments and return values.** For example, open a malicious file using the system call arguments of the “open()” system calls—**arguments and calls.**

Equivalent malicious system calls. For example, after an open(), replace a legitimate read() with a mmap() that reads memory—**false positives.**

Insert some “**no-op**” **system calls** (e.g., read() with 0 byte parameter) between the malicious system call to look like the legitimate sequence—**which calls are they?**

Research Questions

How much code coverage provides a complete learning model to remove/reduce false positives?

How can system call and argument models be efficiently combined?

Are there system calls more important than others?

How much code coverage provides a complete learning model to remove/reduce false positives?



Firefox Dataset

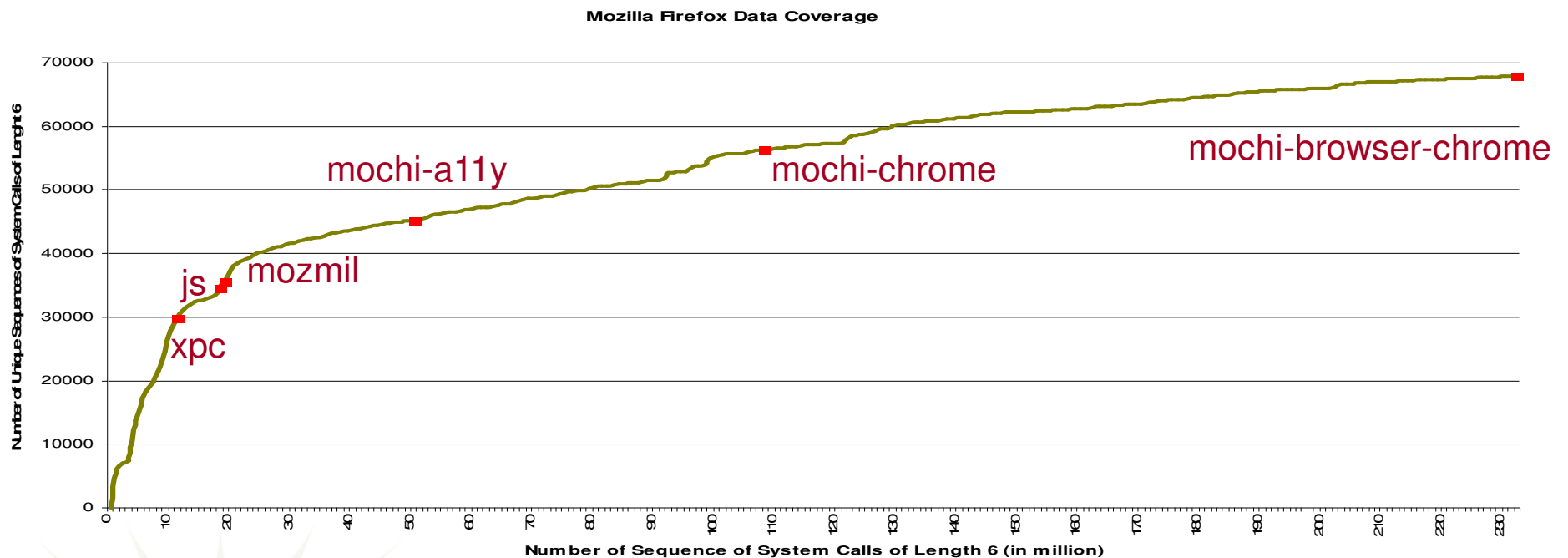
Test Framework	Passing Test Cases	Passing Test Files	Firefox's Code Coverage		Avg. System Calls Per Trace
			Statements (%)	Functions (%)	
XPC Shell	600	600	39.8	39.6	18,479
JS Engine	2686	2686	41.3	40.2	2,534
Mozmill	34	34	47.1	46.1	16,226
Mochitest-a11y	1369	41	49.7	48.7	770,966
Mochitest-chrome	1160	84	50.9	49.6	701,076
Mochitest-browser-chrome	2913	146	52.1	50.6	856,120

System Call Sequence Model

Framework	Total number of sequences in each framework	Total number of unique sequences in each framework	Percentage of unique sequences in each framework
XPC Shell	11,084,489	29,871	0.27%
JS Engine	6,793,621	3,771	0.06%
Mozmill	551,516	1,106	0.20%
Mochitest-a11y	31,609,380	10,440	0.03%
Mochitest-chrome	57,487,806	11,069	0.02%
Mochitest-browser-chrome	124,136,624	11,595	0.01%

Window size = 6.

Impact of Coverage on the Model's completeness

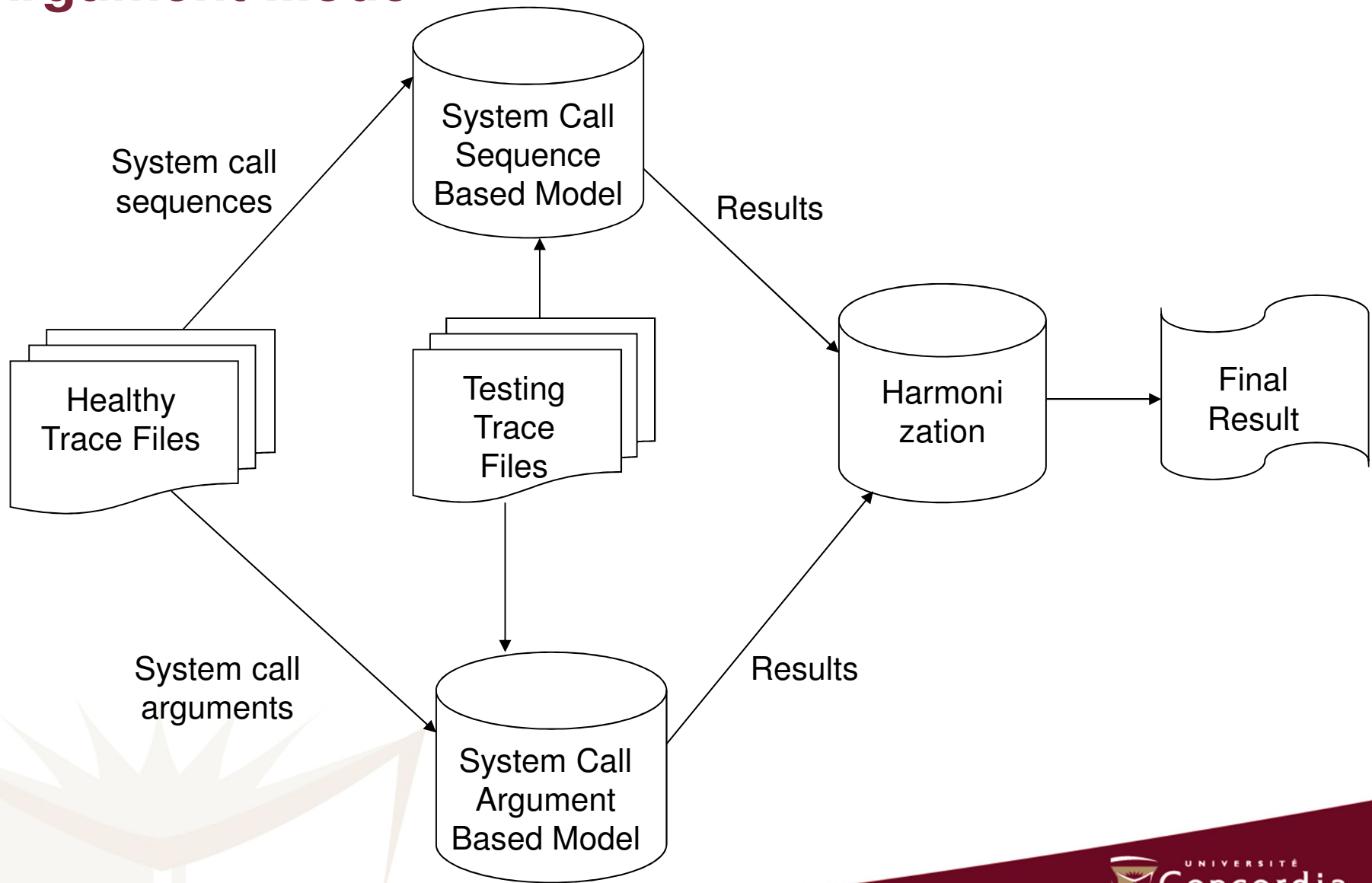


- 67,852 unique sequences identified from 231,663,436 sequences
- Model size is 1,331 KB

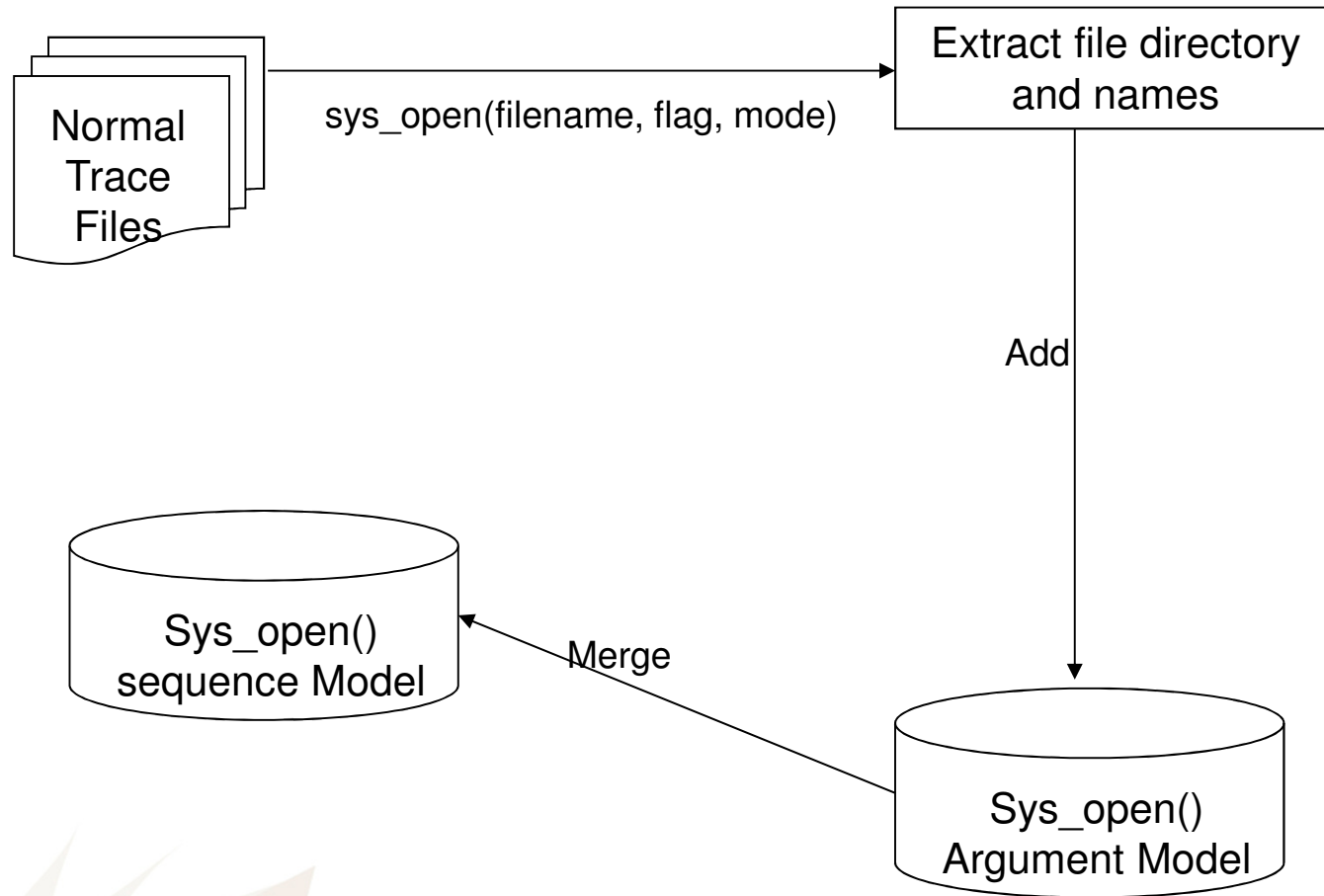
How can system call and argument models be efficiently combined?



Combine System Call Sequence and System Call Argument Model



sys_open() Argument Model Construction



Are there system calls more important than others?



Frequent System Calls

System Call Name	Total Count	Percentage
sys_write	181,115,332	78%
sys_read	9,125,195	4%
sys_llseek	5,669,487	2%
sys_fcntl64	4,358,035	2%
sys_futex	4,209,934	2%
sys_fstat64	4,209,034	2%
sys_stat64	3,603,088	2%
sys_open	2,932,515	1%
sys_gettimeofday	2,897,824	1%
sys_close	2,809,496	1%
sys_madvise	2,742,715	1%
sys_mmap_pgoff	2,612,093	1%
sys_munmap	2,068,300	1%

Thank you & your questions



References

- [Burgess 2002] M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, “Measuring System Normality,” ACM Trans. Computer Systems, vol. 20, no. 2, pp. 125-160, 2002.
- [Bhatkar 2006] S. Bhatkar, A. Chaturvedi, and R. Sekar, “Dataflow Anomaly Detection,” Proc. IEEE Symp. Security and Privacy (S&P ’06), May 2006.
- [Cavallaro 2011] Lorenzo Cavallaro and R. Sekar, “Taint-Enhanced Anomaly Detection”, International Conference on Information System Security (ICISS), December 2001.
- [Cohen 1995] William W. Cohen. 1995. Fast Effective Rule Induction. In Machine Learning: Proceedings of the Twelfth International Conference. Lake Tahoe, California, Morgan Kaufmann.
- [Davis 2002] Davis R. I. A. and Lovell B. C., "Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences", in International Conference on Pattern Recognition, pages 168-171, Quebec City, Canada, August 2002.
- [Denning 1987] Dorothy E. Denning, “An Intrusion-Detection Model”, IEEE Transactions On Software Engineering, Vol. Se-13, No. 2, February 1987, 222-232. (also presented at the 1986 Symp. on Security and Privacy in Oakland, California.) .
- [Feng 2003] Feng H., Kolesnikov O., Fogla P., Lee W. and Gong W. Anomaly Detection Using Call Stack Information. IEEE Symposium on Security and Privacy, 2003.
- [Forrest 1996] S. Forrest, S. A. Hofmeyr, and A. Somayaji. A sense of self for unix processes. In Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1996. IEEE Computer Society Press.
- [Frossi 2009] A Frossi, F. Maggi, G. L. Rizzo, S. Zanero, "Selecting and Improving System Call Models for Anomaly Detection", Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA 2009, Como, Italy, July 2009.
- [Ghosh 1999] Ghosh A., and Schwartzbad A. A Study in Using Neural Networks for Anomaly and Misuse Detection. 1999 USENIX Security Symposium.
- [Jiang 2002] Jiang N., Hua K., and Sheu S. Considering Both Intra-pattern and Inter-pattern Anomalies in Intrusion Detection. Proc. Intl. Conf. Data Mining (ICDM 2002), 2002.
- [Hoang 2003] X. D. Hoang, J. Hu, and P. Bertok, “A Multi-Layer Model for Anomaly Intrusion Detection using Program Sequences of System Calls,” Proc. 11th IEEE Int’l. Conf. Net., Sydney, Australia, Sept. 28–Oct. 1, 2003, pp. 531–36.
- [Hoang 2004] X. D. Hoang and J. Hu, “An Efficient Hidden Markov Model Training Scheme for Anomaly Intrusion Detection of Server Applications Based on System Calls,” IEEE Int’l. Conf. Net. ’04, Singapore, Nov. 16–19, 2004, vol. 2, pp. 470–74.
- [Hofmeyr 1998] S. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion Detection Using Sequences of System Calls,” J. Computer Security, vol. 6, pp. 151-180, 1998.

References (2)

- [Hu 2009] J. Hu, Q. Dong, X. Yu, H.H. Chen: A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection, *IEEE Netw.* 23(1), 42–47 (2009).
- [Jha 2001] S. Jha, K. Tan, and R.A. Maxion, “Markov Chains, Classifiers, and Intrusion Detection,” *Proc. 14th IEEE Workshop Computer Security Foundations (CSFW '01)*, p. 206, 2001.
- [Kang 2005] Kang, D.-K., Fuller, D., and Honavar, V. 2005. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW)*.
- [Kruegel 2003a] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, “On the Detection of Anomalous system Call Arguments,” *Proc. European Symp. Research in Computer Security (ESORICS '03)*, Oct. 2003, pp 101 – 118.
- [Kruegel 2003b] Christopher Kruegel , Darren Mutz , William Robertson , Fredrik Valeur, “Bayesian Event Classification for Intrusion Detection”, *PROCEEDINGS OF ACSAC 2003, LAS VEGAS, NV, 2003*.
- [Lee 1997] Lee W., Stolfo S., and Chan P. Learning Patterns from UNIX Process Execution Traces for Intrusion Detection. *AAAI'97 workshop on AI methods in Fraud and risk management*.
- [Lippmann et al 2000] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. 2000. Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. In *Proceedings of Recent Advances in Intrusion Detection*. LNCS. Springer, Toulouse, France, 162–182.
- [Maggi 2009] F. Maggi, M. Matteucci, S. Zanero. “*Reducing False Positives In Anomaly Detectors Through Fuzzy Alert Aggregation*”. *Information Fusion, special issue on “Information Fusion in Computer Security”*. Vol. 10(4), pp. 300-311 (2009)
- [Maggi 2010] Federico Maggi, Matteo Matteucci And Stefano Zanero, “Detecting Intrusions Through System Call Sequence And Argument Analysis”, *IEEE Transactions On Dependable And Secure Computing*, Vol. 7, No. 4, October-December 2010.
- [Mahoney 2003] Mahoney, M. V., & Chan, P. K.. Learning rules for anomaly detection of hostile network traffic. In *Proc. Of International Conference on Data Mining (ICDM)*, 601-604, 2003.
- [Michael 2012] C. C. Michael And Anup Ghosh, Simple, State-Based Approaches To Program-Based Anomaly Detection, *ACM Transactions on Information and System Security*, Vol. 5, No. 3, August 2002, Pages 203–237.
- [Mutz 2006] Darren Mutz, Fredrik Valeur, Christopher Kruegel, and Giovanni Vigna, “Anomalous System Call Detection”, *ACM Transactions on Information and System Security (TISSEC)*, Volume 9 Issue 1, February 2006.
- [Petrussenko 2010] Denis Petrussenko, Philip K. Chan, “Incrementally Learning Rules for Anomaly Detection”, *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)*.

References (3)

- [Portnoy 2001] Portnoy, L., Eskin, E., and Stolfo, S. 2001. Intrusion Detection with Unlabeled Data using Clustering. In ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001).
- [Sekar 2001] Sekar R., Bendre M., Dhurjati D., Bollineni P. A Fast Automaton-based Method for Detecting Anomalous Program Behaviors. IEEE Symposium on Security and Privacy (S & P), 2001.
- [Snare 2003] SNARE - System iNtrusion Analysis and Reporting Environment.
<http://www.intersectalliance.com/projects/Snare>.
- [Tandon 2003] G. Tandon and P. Chan, “Learning Rules from System Call Arguments and Sequences for Anomaly Detection,” Proc. ICDM Workshop Data Mining for Computer Security (DMSEC '03), pp. 20-29, 2003.
- [Tandon 2007] Gaurav Tandon and Philip K. Chan, “On The Learning Of System Call Attributes For Host-Based Anomaly Detection”, International Journal on Artificial Intelligence Tools, 2007.
- [Wagner 2002] Wagner D., Soto P. Mimicry Attacks on Host-Based Intrusion Detection Systems. ACM Conference on Computer and Communications Security, 2002.
- [Warrender 1999] C. Warrender, S. Forrest, and B.A. Pearlmutter, “Detecting Intrusions Using System Calls: Alternative Data Models,” Proc. 1999 IEEE Symposium on Security and Privacy (S&P '99), IEEE Computer Society, pp. 133-145, 1999.
- [Wespi 1999] Wespi A., Dacier M., and Debar H. An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm. Proc. EICAR, 1999.
- [Xu 2006] Xu, W., Bhatkar, S., Sekar, R.: Taint-enhanced Policy Enforcement: a Practical Approach to Defeat a Wide Range of Attacks. In: USENIX Security Symposium (2006)
- [Ye 2001] N. Ye and Q. Chen, “An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems,” Quality and Reliability Eng. Int'l, vol. 17, no. 2, pp. 105-112, 2001.

References (4)

- [Burgess 2002] M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, “Measuring System Normality,” ACM Trans. Computer Systems, vol. 20, no. 2, pp. 125-160, 2002.
- [Bhatkar 2006] S. Bhatkar, A. Chaturvedi, and R. Sekar, “Dataflow Anomaly Detection,” Proc. IEEE Symp. Security and Privacy (S&P ’06), May 2006.
- [Cavallaro 2011] Lorenzo Cavallaro and R. Sekar, “Taint-Enhanced Anomaly Detection”, International Conference on Information System Security (ICISS), December 2001.
- [Cohen 1995] William W. Cohen. 1995. Fast Effective Rule Induction. In Machine Learning: Proceedings of the Twelfth International Conference. Lake Tahoe, California, Morgan Kaufmann.
- [Davis 2002] Davis R. I. A. and Lovell B. C., "Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences", in International Conference on Pattern Recognition, pages 168-171, Quebec City, Canada, August 2002.
- [Denning 1987] Dorothy E. Denning, “An Intrusion-Detection Model”, IEEE Transactions On Software Engineering, Vol. Se-13, No. 2, February 1987, 222-232. (also presented at the 1986 Symp. on Security and Privacy in Oakland, California.) .
- [Feng 2003] Feng H., Kolesnikov O., Fogla P., Lee W. and Gong W. Anomaly Detection Using Call Stack Information. IEEE Symposium on Security and Privacy, 2003.
- [Forrest 1996] S. Forrest, S. A. Hofmeyr, and A. Somayaji. A sense of self for unix processes. In Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1996. IEEE Computer Society Press.
- [Frossi 2009] A Frossi, F. Maggi, G. L. Rizzo, S. Zanero, "Selecting and Improving System Call Models for Anomaly Detection", Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA 2009, Como, Italy, July 2009.
- [Ghosh 1999] Ghosh A., and Schwartzbad A. A Study in Using Neural Networks for Anomaly and Misuse Detection. 1999 USENIX Security Symposium.
- [Jiang 2002] Jiang N., Hua K., and Sheu S. Considering Both Intra-pattern and Inter-pattern Anomalies in Intrusion Detection. Proc. Intl. Conf. Data Mining (ICDM 2002), 2002.
- [Hoang 2003] X. D. Hoang, J. Hu, and P. Bertok, “A Multi-Layer Model for Anomaly Intrusion Detection using Program Sequences of System Calls,” Proc. 11th IEEE Int’l. Conf. Net., Sydney, Australia, Sept. 28–Oct. 1, 2003, pp. 531–36.
- [Hoang 2004] X. D. Hoang and J. Hu, “An Efficient Hidden Markov Model Training Scheme for Anomaly Intrusion Detection of Server Applications Based on System Calls,” IEEE Int’l. Conf. Net. ’04, Singapore, Nov. 16–19, 2004, vol. 2, pp. 470–74.
- [Hofmeyr 1998] S. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion Detection Using Sequences of System Calls,” J. Computer Security, vol. 6, pp. 151-180, 1998.

References (5)

- [Hu 2009] J. Hu, Q. Dong, X. Yu, H.H. Chen: A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection, *IEEE Netw.* 23(1), 42–47 (2009).
- [Jha 2001] S. Jha, K. Tan, and R.A. Maxion, “Markov Chains, Classifiers, and Intrusion Detection,” *Proc. 14th IEEE Workshop Computer Security Foundations (CSFW '01)*, p. 206, 2001.
- [Kang 2005] Kang, D.-K., Fuller, D., and Honavar, V. 2005. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW)*.
- [Kruegel 2003a] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, “On the Detection of Anomalous system Call Arguments,” *Proc. European Symp. Research in Computer Security (ESORICS '03)*, Oct. 2003, pp 101 – 118.
- [Kruegel 2003b] Christopher Kruegel , Darren Mutz , William Robertson , Fredrik Valeur, “Bayesian Event Classification for Intrusion Detection”, *PROCEEDINGS OF ACSAC 2003, LAS VEGAS, NV, 2003*.
- [Lee 1997] Lee W., Stolfo S., and Chan P. Learning Patterns from UNIX Process Execution Traces for Intrusion Detection. *AAAI'97 workshop on AI methods in Fraud and risk management*.
- [Lippmann et al 2000] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. 2000. Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. In *Proceedings of Recent Advances in Intrusion Detection*. LNCS. Springer, Toulouse, France, 162–182.
- [Maggi 2009] F. Maggi, M. Matteucci, S. Zanero. “*Reducing False Positives In Anomaly Detectors Through Fuzzy Alert Aggregation*”. *Information Fusion, special issue on “Information Fusion in Computer Security”*. Vol. 10(4), pp. 300-311 (2009)
- [Maggi 2010] Federico Maggi, Matteo Matteucci And Stefano Zanero, “Detecting Intrusions Through System Call Sequence And Argument Analysis”, *IEEE Transactions On Dependable And Secure Computing*, Vol. 7, No. 4, October-December 2010.
- [Mahoney 2003] Mahoney, M. V., & Chan, P. K.. Learning rules for anomaly detection of hostile network traffic. In *Proc. Of International Conference on Data Mining (ICDM)*, 601-604, 2003.
- [Michael 2012] C. C. Michael And Anup Ghosh, Simple, State-Based Approaches To Program-Based Anomaly Detection, *ACM Transactions on Information and System Security*, Vol. 5, No. 3, August 2002, Pages 203–237.
- [Mutz 2006] Darren Mutz, Fredrik Valeur, Christopher Kruegel, and Giovanni Vigna, “Anomalous System Call Detection”, *ACM Transactions on Information and System Security (TISSEC)*, Volume 9 Issue 1, February 2006.
- [Petrussenko 2010] Denis Petrussenko, Philip K. Chan, “Incrementally Learning Rules for Anomaly Detection”, *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)*.

References (6)

- [Portnoy 2001] Portnoy, L., Eskin, E., and Stolfo, S. 2001. Intrusion Detection with Unlabeled Data using Clustering. In ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001).
- [Sekar 2001] Sekar R., Bendre M., Dhurjati D., Bollineni P. A Fast Automaton-based Method for Detecting Anomalous Program Behaviors. IEEE Symposium on Security and Privacy (S & P), 2001.
- [Snare 2003] SNARE - System iNtrusion Analysis and Reporting Environment.
<http://www.intersectalliance.com/projects/Snare>.
- [Tandon 2003] G. Tandon and P. Chan, “Learning Rules from System Call Arguments and Sequences for Anomaly Detection,” Proc. ICDM Workshop Data Mining for Computer Security (DMSEC '03), pp. 20-29, 2003.
- [Tandon 2007] Gaurav Tandon and Philip K. Chan, “On The Learning Of System Call Attributes For Host-Based Anomaly Detection”, International Journal on Artificial Intelligence Tools, 2007.
- [Wagner 2002] Wagner D., Soto P. Mimicry Attacks on Host-Based Intrusion Detection Systems. ACM Conference on Computer and Communications Security, 2002.
- [Warrender 1999] C. Warrender, S. Forrest, and B.A. Pearlmutter, “Detecting Intrusions Using System Calls: Alternative Data Models,” Proc. 1999 IEEE Symposium on Security and Privacy (S&P '99), IEEE Computer Society, pp. 133-145, 1999.
- [Wespi 1999] Wespi A., Dacier M., and Debar H. An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm. Proc. EICAR, 1999.
- [Xu 2006] Xu, W., Bhatkar, S., Sekar, R.: Taint-enhanced Policy Enforcement: a Practical Approach to Defeat a Wide Range of Attacks. In: USENIX Security Symposium (2006)
- [Ye 2001] N. Ye and Q. Chen, “An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems,” Quality and Reliability Eng. Int'l, vol. 17, no. 2, pp. 105-112, 2001.

References (7)

- [Baliga et al., 2008] Baliga, A., Iftode, L., Chen, X.: Automated containment of rootkits attacks. *Computers and Security* 27(7-8), 2008, pp. 323–334 .
- [Baliga et al., 2011] Baliga, A., Ganapathy, V., and Iftode, L. Detecting kernel-level rootkits using data structure invariants. *IEEE Transactions on Dependable and Secure Computing* 8, 5 (2011), pp. 670 –684.
- [Bauer 2006] Bauer M. Paranoid penguin: an introduction to Novell AppArmor. *Linux J* 2006; 148-161
- [Beck et al, 2005] Beck, D., Vo, B., Verbowski, C.: Detecting stealth software with strider ghostbuster. In: *Proceedings of the 2005 International Conference on Dependable Systems and Networks, DSN 2005*, pp. 368–377.
- [Bratus et al., 2010] Bratus, S., Locasto, M.E., Ramaswamy, A., Smith, S.W.: Vm-based security overkill: a lament for applied systems security research. In: *Proceedings of the 2010 Workshop on New Security Paradigms, NSPW 2010*, pp. 51–60. ACM, New York, 2010.
- [Carbone et al., 2009] Carbone, M.; Cui, W.; Lu, L., Lee, W., Peinado, M., Jiang, X.. Mapping kernel objects to enable systematic integrity checking. In: *Proc. 16th ACM Conference on Computer and Communications Security*. 2009.
- [DARPA Technical Report, 2007] DARPA: Rootkit Detection, draft v2.10, April 26, 2007: <http://www.scribd.com/doc/53451398/6/Rootkit-Classification>

References (8)

- [Garfinkel and Rosenblum, 2003] Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proc. Network and Distributed Systems Security Symposium, vol. 1, pp. 253–285. Citeseer, 2003.
- [Jiang et al., 2007] Jiang, X., Wang, X., Xu, D.: Stealthy malware detection through vm-based out of the box semantic view reconstruction, pp. 128–138, 2007.
- [Jaeger et al., 2011] Jaeger, T.; Van Oorschot, P. C.; Wurster, G.; “Countering Unauthorized Code Execution on Commodity Kernels: A Survey of Common Interfaces Allowing Kernel Code Modification”, 2011
- [Jestin et al., 2011b] Jestin, J.; Anita, J.; James, J. “Rootkit Detection Mechanism: A Survey”, *Advances in Parallel Distributed Computing, Communications in Computer and Information Science, Springer, Vol. 203*, 2011, pp. 366-374
- [Jestin et al., 2011a] Jestin, J.; Anita, J.; “A Host Based Kernel Level Rootkit Detection Mechanism Using Clustering Technique”, *Trends in Computer Science, Engineering and Information Technology, Communications in Computer and Information Science, Springer, 2011, Vol. 204*, pp: 564-570

References

- [Kroah-Hartman, 2004] Kroah-Hartman, G.: Signed kernel modules. Linux Journal, 2004.
- [Kruegel et al., 2004] Kruegel, C., Robertson, W., Vigna, G.: Detecting kernel-level rootkits through binary analysis. In: Computer Security Applications Conference, Annual, pp. 91–100, 2004
- [Kim & Spafford, 1994] Kim, G.H.; Spafford, E.H.: The design and implementation of tripwire: a file system integrity checker. In: Proceedings of the 2nd ACM Conference on Computer and Communications Security, CCS 1994, pp. 18–29. ACM, New York (1994),
- [Lanzi et al., 2009] Lanzi, A., Sharif, M., Lee, W.: K-tracer: A system for extracting kernel malware behavior. In: Proceedings of the 16th Annual Network and Distributed System Security Symposium, 2009.
- [Levin et al., 2005]Levine, J., Grizzard, J., Owen, H.: A methodology to detect and characterize kernel level rootkit exploits involving redirection of the system call table. In: Proceedings of Second IEEE International Information Assurance Workshop, pp. 107–125. IEEE, Los Alamitos (2005)
- [McAfee Whitepaper , 2009] 2010 threat predictions, December 2009. McAfee AVERT Labs Whitepaper.

References

- [McAfee, Whitepaper 2006] Rootkits, part 1 of 3: A growing threat, April 2006. McAfee AVERT Labs Whitepaper.
- [Parno et al., 2010] Parno, B; McCune JM; Perrig A. “ Bootstrapping trust in commodity computers. In Proc: 2010 IEEE symposium on security and privacy; 2010, p. 414-429
- [Pelaez, 2004] Pelaez, R., S.; *Linux kernel rootkits: protecting the system’s “Ring-Zero”*, May, 2004. http://www.sans.org/reading_room/whitepapers/honors/linux-kernel-rootkits-protecting-systems_1500
- [Petroni et al., 2004] Petroni Jr., N.L., Fraser, T., Molina, J., Arbaugh, W.A.: Copilot - a coprocessor based kernel runtime integrity monitor. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, vol. 13, pp. 13–13. USENIX Association, Berkeley (2004).
- [Petroni et al., 2007] Petroni Jr., N.L., Hicks, M.: Automated detection of persistent kernel control-flow attacks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 103–115.
- [Rutkowska, 2006] Joanna Rutkowska, Introducing Stealth Malware Taxonomy, COSEINC Advanced Malware Labs, Nov., 2006. <http://www.scribd.com/doc/55716332/Malware-Taxonomy>

References

- [Rhee et al., 2009] Rhee, J., Riley, R., Xu, D., Jiang, X.: Defeating dynamic data kernel rootkit attacks via vmm-based guest-transparent monitoring. In: International Conference on Availability, Reliability and Security, pp. 74–81 (2009)
- [Riley et al., 2008] Riley, R., Jiang, X., Xu, D.: Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 1–20. Springer, Heidelberg (2008)
- [Riley et al., 2009] Riley, R., Jiang, X., Xu, D.: Multi-aspect profiling of kernel rootkit behavior. In: Proceedings of the 4th ACM European Conference on Computer Systems, EuroSys2009, pp. 47–60. ACM, New York (2009),
- [Smalley et al., 2002] Smalley S, Vance C, Salamon W. “Implementing SELinux as a linux security module. Technical Report 01-043. NAI Labs; 2002.
- [Sailer et al., 2004] Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a tcg-based integrity measurement architecture. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, vol. 13, p. 16. USENIX Association, Berkeley (2004),

References

- [Seshadri et al., 2007] Seshadri, A., Luk, M., Qu, N., Perrig, A.: Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In: Proceedings of the 21st ACM Symposium on Operating Systems Principles (21st SOSP 2007), pp. 335–350. ACM SIGOPS, Stevenson (October 2007)
- [Wang et al., 2009] Wang, Z., Jiang, X., Cui, W., and Ning, P. Countering kernel rootkits with lightweight hook protection. In Proceedings of the ACM Conference on Computer and Communications Security (2009), pp. 545–554.
- [Wichiman, 2009]Wichmann, R.: A comparison of several host/file integrity monitoring programs, December 29, 2009.
- [Yin et al., 2008] Yin, H., Liang, Z., Song, D.: Hookfinder: Identifying and understanding malware hooking behaviors. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS 2008).

Resources for Rootkits

- <http://www.phrack.org>
- <http://www.antiserver.it/Backdoor-Rootkit/>
- <http://www.l0t3k.org/tools/Rootkit/>
- <http://packetstormsecurity.org/UNIX/penetration/rootkits/>
- <http://www.securityfocus.com/>
- <http://www.antiserver.it/Backdoor-Rootkit/>
- <http://www.zone-h.org/en/download/category=23/>
- <http://www.rootkit.com> (mostly Windows based)
- <http://www.blackhat.com/html/bh-media-archives/bh-multi-media-archives.html>

Disclaimer: Presenter doesn't take responsibility of any malicious activity that would happen on your system when you play with these resources ☺.

Source: [Pelaez, 2004]