Improving Host Based Anomaly Detection

Shariyar, Afroza, Prasanna and Abdelwahab

Software Behaviour Analysis Research Lab

abdelw@ece.concordia.ca

Montreal, QC

Dec. 9th , 2011

Reporting on Four Studies that We Conducted: Outline

- Our objective is to investigate advanced anomaly detection techniques
  - Study 1 - Comparing kernel space and user space tracing mechanisms for anomaly detection
  - Study 2 - Reducing false positive rate using generalization of system calls
  - Study 3 - Enhanced hidden Markov model using the concept of n-gram
  - Study 4 - Linux-based attack taxonomy

Comparison of user space and kernel space traces in discovering anomalous software behaviour

Duration: May to July and October, 2011

Why Identify Anomalous Behaviour?

- Identification of normal and anomalous software behaviour is important in:
  - Software debugging, such as fault localization (Murtaza et al. 2010, Jones et al. 2005)
    User space tracing
  - Autonomic computing, such as self managing applications (Jiang et al. 2005)
    User space tracing
  - Software intrusion, such as anomaly detection systems (Warrender et al. 1999, Wang et al. 2004)
    Kernel space tracing

Trace Examples

Function call trace (User space)

```
23    fooPrevious exit
24     foo1 entry
25
26
27
28
29     | | foo3 exit
30     | foo2 exit
31     foo1 exit
32    fooLater entry
```

System events trace (Kernel space)

channel:kernel; event:syscall_entry process:./gzip.exe; state: SYSCALL; markers:ip = 0x22cbad, syscall_id = 6 [sys_close+0x0/0x100]; pid:2842

channel:fs; event:close process:./gzip.exe; state:SYSCALL;

x0/
0x70]

channel:kernel; event:irq_exit process:./gzip.exe; state:USER_MODE; markers:handled = 1; pid:2842

channel:kernel; event:page_fault_entry process:./gzip.exe; state:TRAP; markers:ip = 0x8049aa9, address = 0x805d000, trap_id = 14, write_access = 1; pid:2842

No comparison of user space and kernel space tracing exists in the literature: can we substitute one with another or which one is the best?

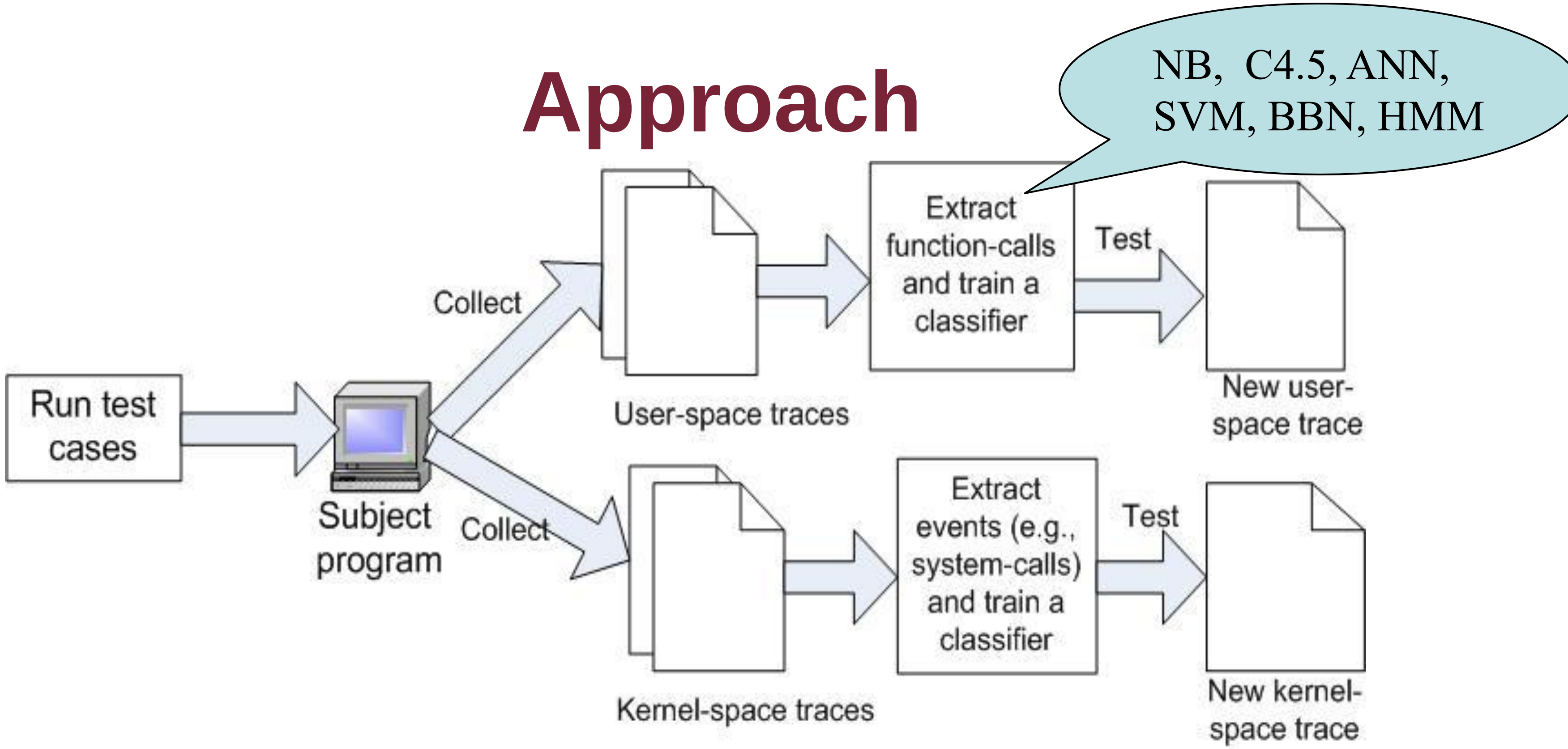
Research Questions

- *(Q1) Can kernel space tracing be used to classify pass fail traces of a program with the same accuracy as user space tracing?*

- To find the answer we employed six classification algorithms (i.e., NB, C4.5, ANN, SVM, BBN, and HMM) and in the process identified a novel secondary research question.

- *(Q2) Can we substitute one classification algorithm with another without affecting the accuracy of classification of normal and abnormal traces?*

Approach



- Step 1: Collect user space and kernel space traces.
- Step 2: Extract events (e.g., function or system calls).
- Step 3: For each type of tracing evaluate all the classifiers from two perspectives:
  - (a) Training and testing on both normal and anomalous traces.
  - (b) Training on only normal traces and testing on both types of traces.

Dataset

Releases used: Flex 2.5.1; Grep 2.4; Gzip 1.1.2; Sed 4.0.7.					
Prog.	LOC	# Functions	# Faults	# Passing Traces	#Failed Traces
Flex	9724	167	20	566	545
Grep	9041	149	18	799	710
Gzip	4032	88	16	214	204
Sed	4735	115	6	366	166

Results for two-class classification

Results on user space traces

	Flex		
Algo.	TP	FP	AUC
C4.5	0.924	0.099	0.925
NB	0.159	0.053	0.609
BBN	0.371	0.145	0.675
ANN	0.981	0.804	0.646
SVM	0.721	0.323	0.699
HMM	0.706	0.0	0.416

Results on kernel space traces

	Flex		
Algo.	TP	FP	AUC
C4.5	1.00	0.002	0.998
NB	1.00	0.002	0.999
BBN	0.993	0.004	0.999
ANN	0.998	0.002	1.00
SVM	0.998	0.002	0.998
HMM	1.00	0.002	0.996

Answers to Research Questions

(1) Kernel space tracing identifies software anomalies better than the function call traces at user space level

- Time to train classifiers on kernel space traces was 20-60% less than user space traces

(2) No significant difference exist among classification algorithms in detection of software anomalies using execution traces

- However, the C4.5 decision tree yields higher accuracy in two-class classification and neural network yields higher accuracy in one-class classification.

Reduction of false positive rate in anomaly detection through generalization of system calls.

Duration: Aug. to Oct., 2011.

False positives: A major problem in anomaly detection system

- A major problem is the generation of number of incorrect alarms on normal software behaviour— i.e., false positives.
- A large number of false positives in anomaly detection systems have made the misuse (signature based) detection systems first choice in the industry.

Is the problem in the application of algorithms on different datasets or is it in the properties of underlying data?

Motivating Example

Sequence 1
fork
read
read
fork
read
read
fork
read
read
fork
read
read

Sequence 2
fork
read
read
fork
read
read
fork
read
fork
read

Different contiguous repetitions of system calls but the task performed is exactly the same: **creation of a process (fork) and reading from an I/O device (read).**

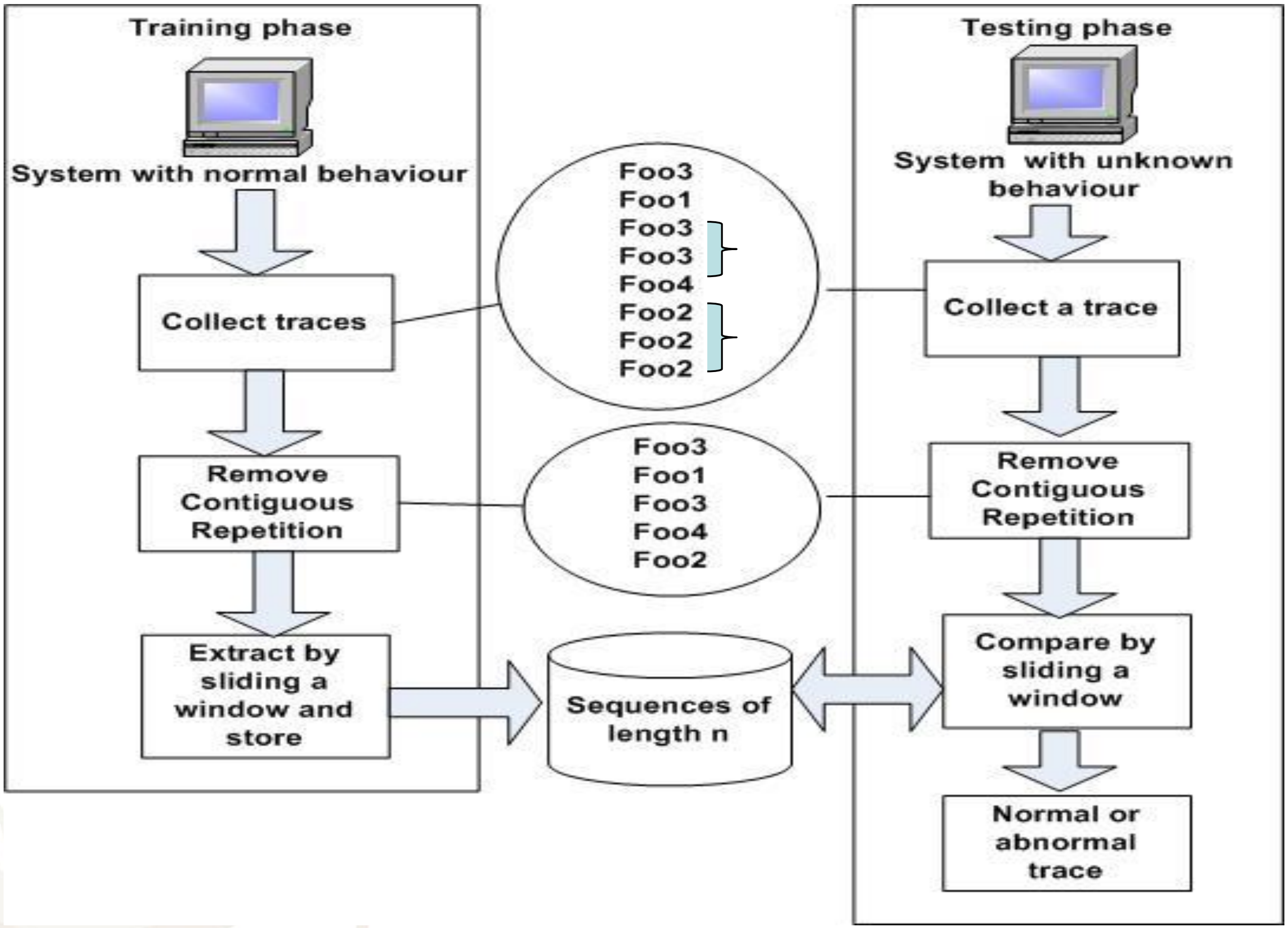
There will be a mismatch (false positive) for “Sequence 2” if an algorithm is trained on “Sequence 1”, even though the task is the same.

These observations warrant an empirical investigation.

Research Hypothesis

On generalizing system calls, we can reduce false positive rate of an anomaly detection algorithm without affecting the true positive rate

Approach



Training phase
System with normal behaviour
Collect traces
Remove Contiguous Repetition
Extract by sliding a window and store
Foo3
Foo1
Foo3
Foo3
Foo4
Foo2
Foo2
Foo2
Foo3
Foo1
Foo3
Foo4
Foo2
Sequences of length n
Testing phase
System with unknown behaviour
Collect a trace
Remove Contiguous Repetition
Compare by sliding a window
Normal or abnormal trace

Datasets

Program	Intrusion traces	Normal traces	Normal traces used for training	Normal traces used for testing
Sendmail	25	346	135	211
Stide	105	13726	600	13126
MIT live lpr	1001	2703	415	2288
UNM live lpr	1001	4298	390	3908
Xlock	2	1731	121	1610

Results

		Sendmail		Stide		MIT live lpr		UNM live lpr		Xlock	
		FP	TP	FP	TP	FP	TP	FP	TP	FP	TP
Win(w) = 6	Stide	24	16	69	104	196	1001	571	1001	24	2
	CRA	23	16	66	104	181	1001	327	1001	18	2
Win(w)= 10	Stide	27	16	12746	104	350	1001	803	1001	24	2
	CRA	25	16	137	104	183	1001	356	1001	18	2
Win(w)=15	Stide	30	16	12760	104	458	1001	869	1001	24	2
	CRA	27	16	187	105	183	1001	423	1001	18	2
Win(w)=20	Stide	33	18	12770	104	537	1001	958	1001	24	2
	CRA	33	18	188	105	212	1001	473	1001	18	2

Significant difference exists in false positives, according to Wilcoxon signed rank test; but no significant increase in TP at higher win width.

Results

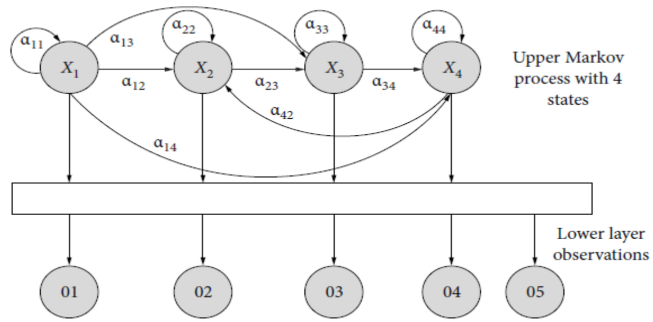
- At window width 6, the effect size is 0.5482 between our approach and Stide:
  - The results are interpreted as:
    - The average false positive rate of Stide will be 0.5482 standard deviations above than the average false positive rate of CRA.

- **Thus, our hypothesis has been validated: False positives can be reduced significantly by removing contiguous repetitions of system calls.**

Enhanced hidden Markov model using the concept of n-gram.

Duration: May to Oct., 2011.

Hidden Markov Model (HMM)



Advantage:
Very Accurate

Disadvantage:
Very Slow

The Time Complexity = O(N(1+T(M+N))) [Langford 07]
- Number of Observables (M)
- Number of Hidden States (N)
- Length of Training Sequences (T)

Proposed Algorithm: I-HMM

HMM

Reduce the training time

- Reduce the number of observables
- Reduce the length of training sequences

Use frequent pattern (N-Gram) extracting technique

I-HMM

N-Gram Extraction: Example

Training Traces: ECDB, CDBA and EACDB

1-gram	2-gram	3-gram
A (2)	~~EC (1)~~	CDB (3)
B (3)	CD (3)	
C (3)	DB (3)	
D (3)	~~BA (1)~~	
E (2)	~~EA (1)~~	
	~~AC (1)~~	

$$\text{frequency}(p_{k+1}) > \alpha * \min(\text{frequency}(q_k), \text{frequency}(r_k))$$

$[\alpha = 0.6]$

Replacement of N-Grams: Example

Training Traces: ECDB, CDBA and EACDB

Assign unique ID: A(2) = 1, B(3) = 2, C(3) = 3, D(3) = 4, E(2) = 5, CD(3) = 6, DB(3) = 7, CDB(3) = 8

ECDB, CDBA, EACDB

Replace CDB

E8, 8A, EA8

Replace A

E8, 81, E18

Replace E

58, 51, 518

Model Construction

HMM

- Set of Observables = {A, B, C, D, E}
- Set of Training Sequences = {ECDB, CDBA, EACDB}
- Set of Hidden States = {X1, X2, …., Xm}

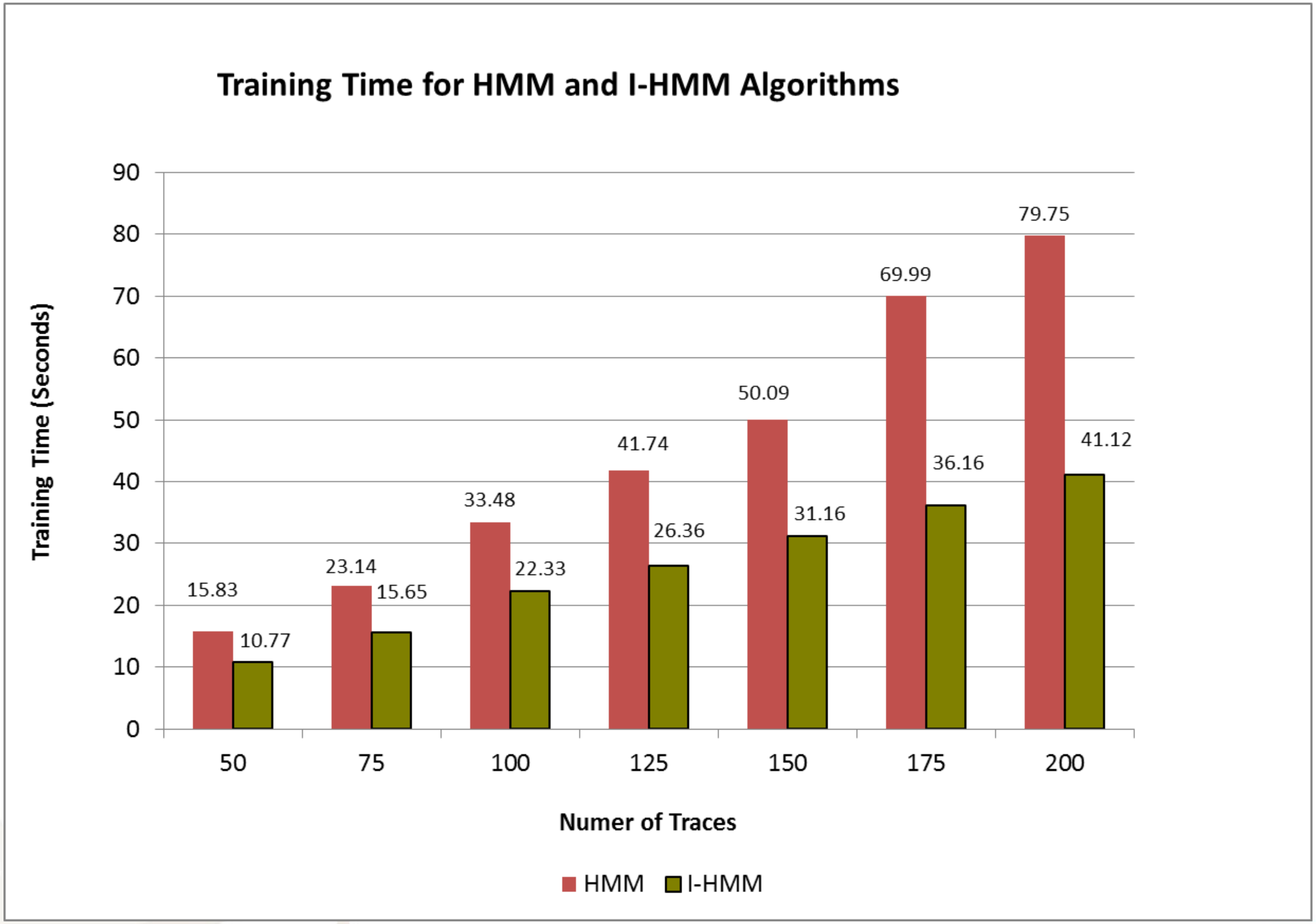
I-HMM

- Set of Observables = {1, 5, 8}
- Set of Training Sequences = {58, 51, 518}
- Set of Hidden States = {X1, X2, …., Xm}

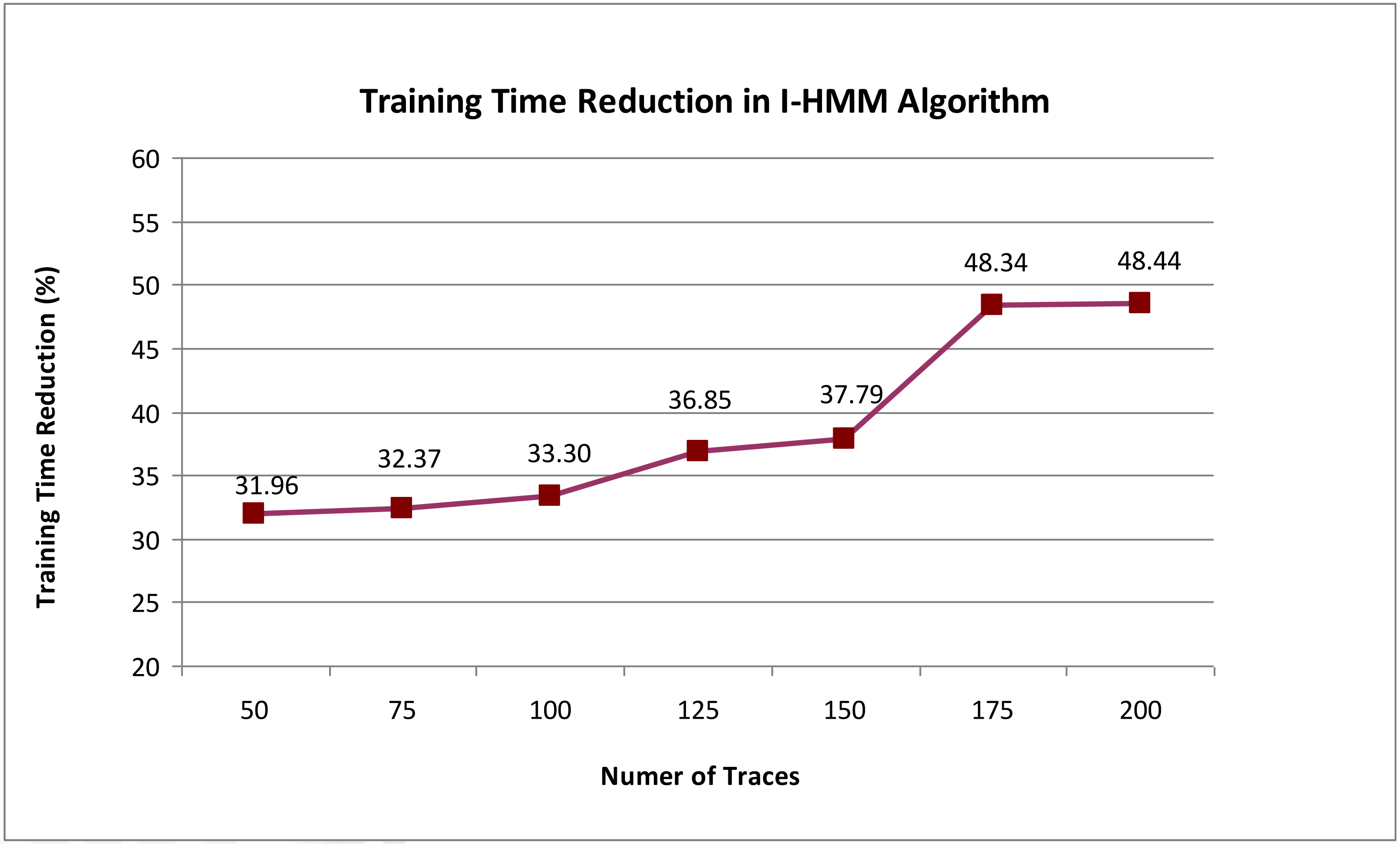
Reduces the size of the set of the observables

Reduces the length of the training sequences.

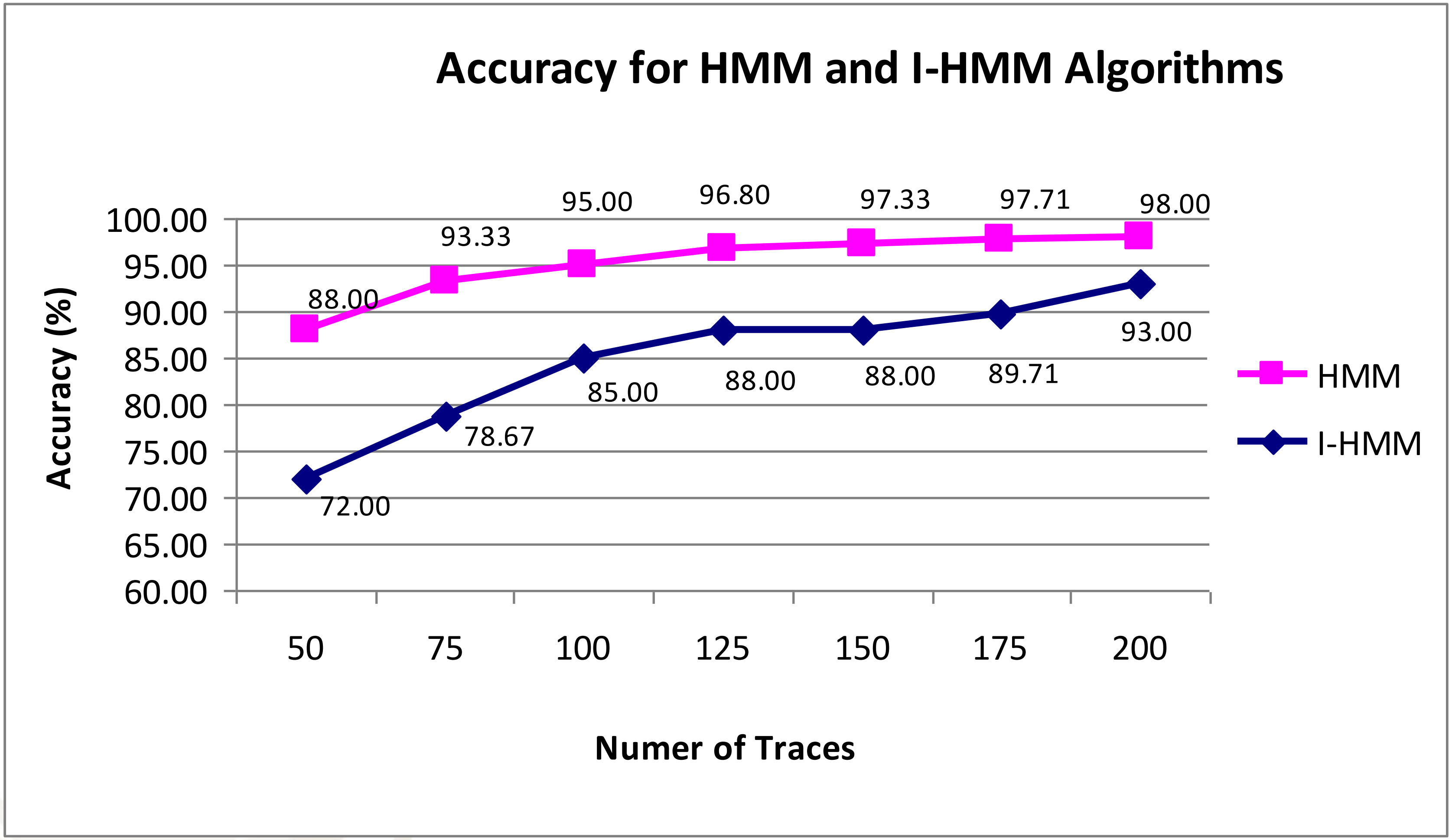
Experiments and Results



Experiments and Results



Experiments and Results



Linux Kernel-based attack taxonomy.

Duration: Sep. to Nov., 2011

Objective

- Build **a taxonomy** of known attacks and vulnerabilities for **the Linux kernel**
  - That can lead to techniques for mitigating these attacks
- There exist many attack taxonomies
  - They vary in coverage and target platforms
  - **None focuses explicitly on the Linux kernel**
  - Refer to: “AVOIDIT: A Cyber Attack Taxonomy” by C. Simmons et al. from the University of Memphis

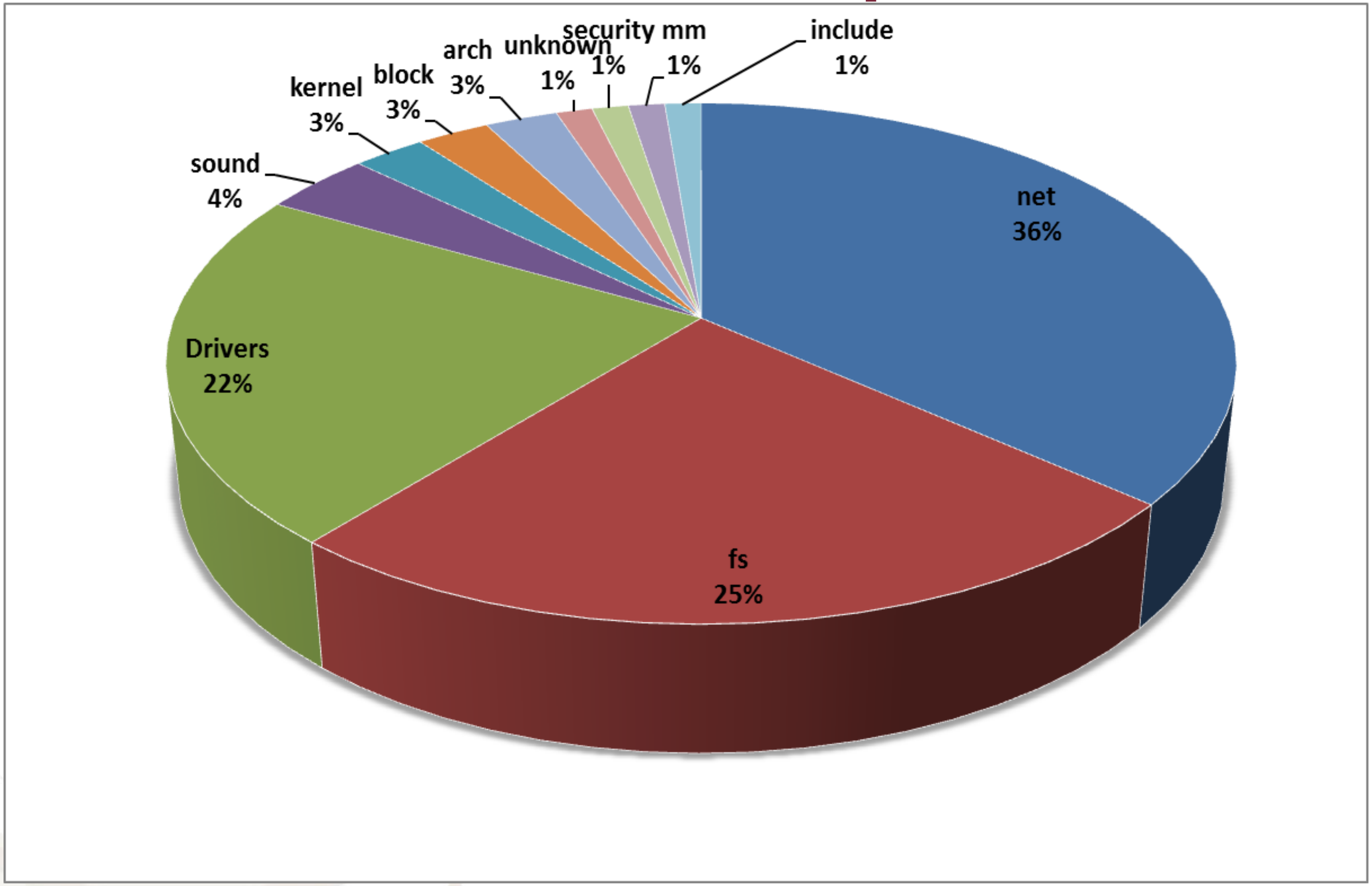
Proposed Attack Taxonomy Framework

•**Affected component**	•The component of the Linux kernel that is vulnerable: Net, fs, etc..
•**Effect of the attack**	•What effect the attack has on the system: DoS, privilege escalation, information disclosure
•**Origin of attack**	•Locally exploitable, local area network exploitable, and remotely exploitable
•**Complexity of access**	•The need of privileges, special conditions, presence of other vulnerabilities...
•**Impact**	•Classified into confidentiality, Integrity, and Availability

Analysis of the 2011 Linux Kernel Vulnerabilities

- We studied 77 vulnerabilities in the Linux Kernel discovered and reported in the year 2011
  - Based on the vulnerabilities discovered and reported in 2011
  - We used www.cvedetails.com to filter Linux Kernel based vulnerabilities from the CVE database

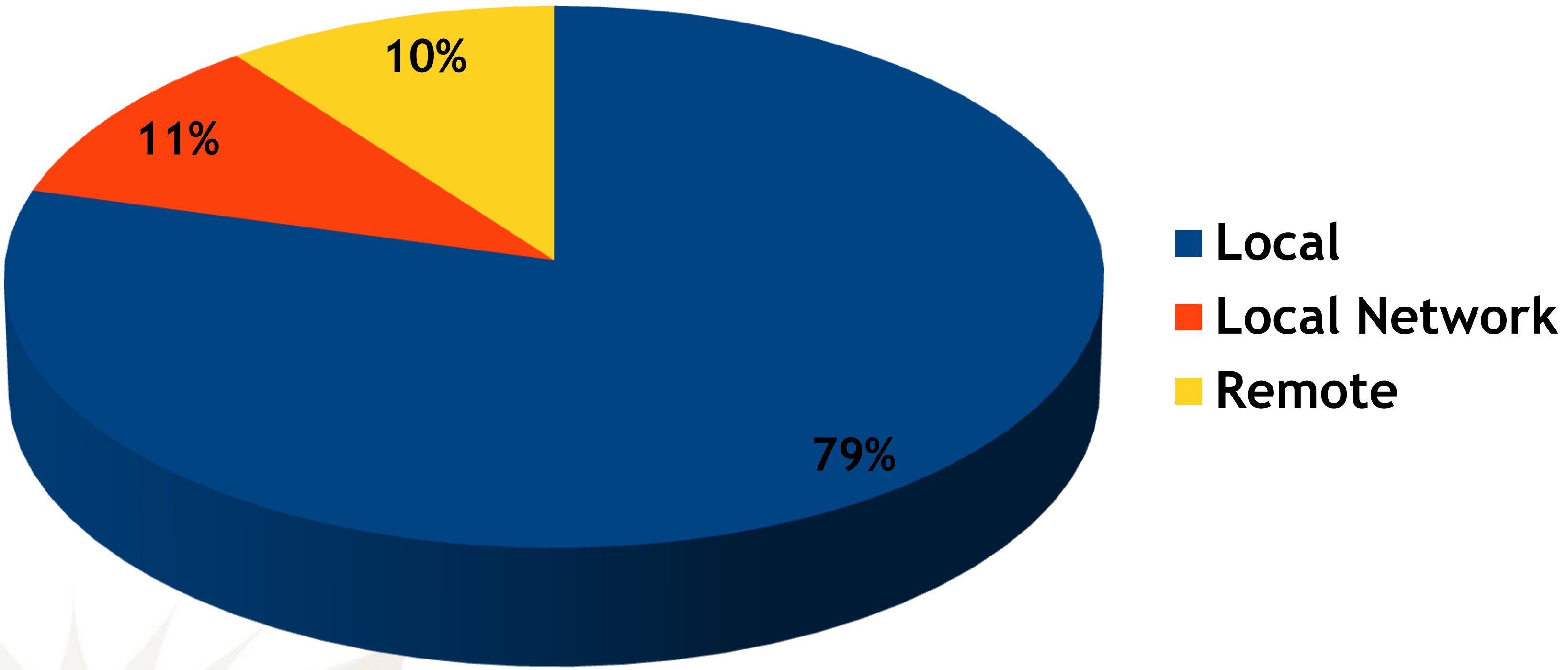
Analysis of 2011 Kernel Vulnerabilities Based on Affected Component



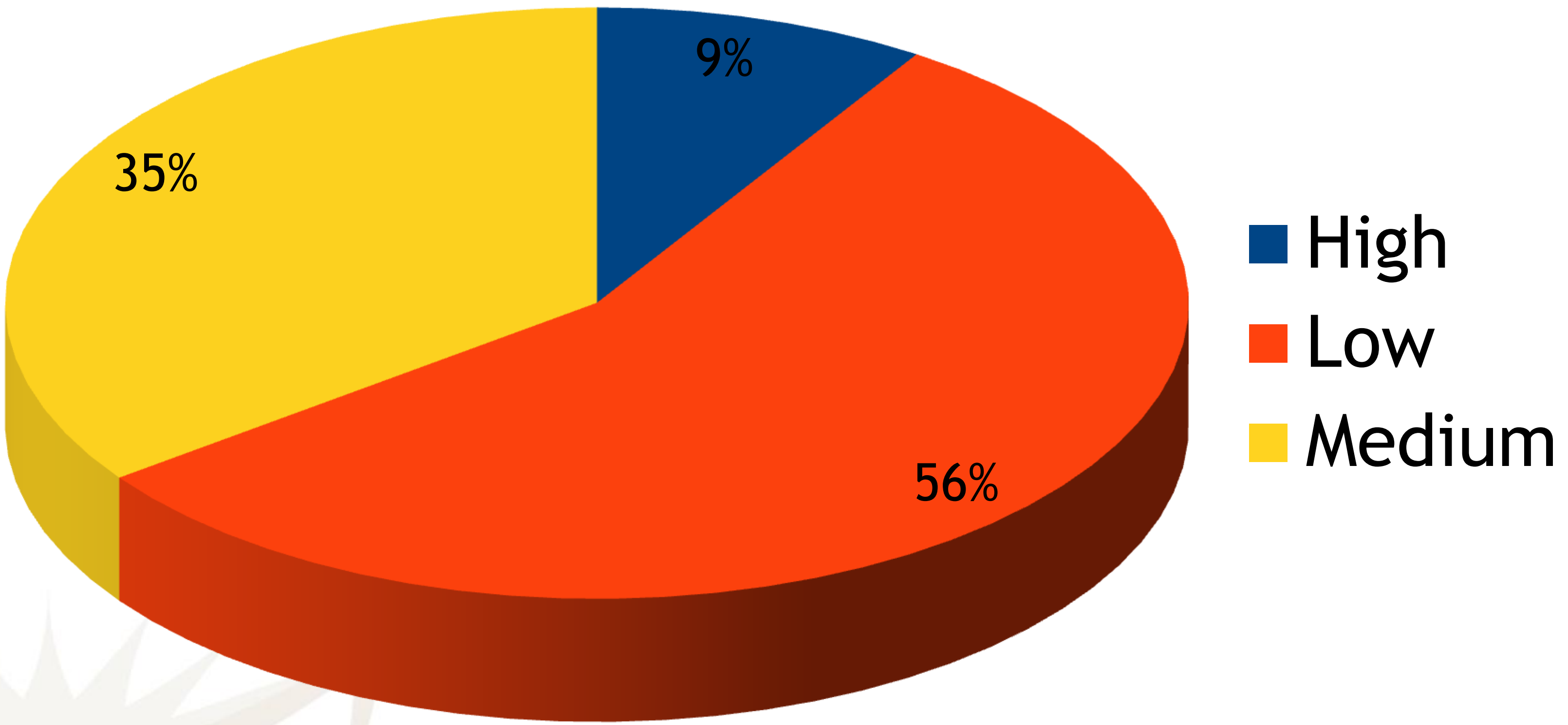
Analysis of 2011 Kernel Vulnerabilities Based on the Effect of Attack

Obtain Sensitive Information	15
Denial of Service/Overflow	8
Denial of Service/Overflow/Memory Corruption	4
Denial of Service/Memory Corruption	2
Denial of Service/Overflow/Gain Privileges/Memory Corruption	2
Overflow/Gain Privileges/Obtain Sensitive Information	2
Unspecified	2
Denial of Service/Bypass	1
Denial of Service/Gain Privileges	1
Denial of Service/Gain Privileges/Memory Corruption	1
Denial of Service/Gain Privileges/Memory Corruption/Obtain Sensitive Information	1
Denial of Service/Obtain Sensitive Information	1
Denial of Service/Overflow/Gain Privileges	1
Denial of Service/Overflow/Obtain Sensitive Information	1
Denial of Service/ Overflow/ Gain Privileges	1
Overflow/Obtain Sensitive Information	1

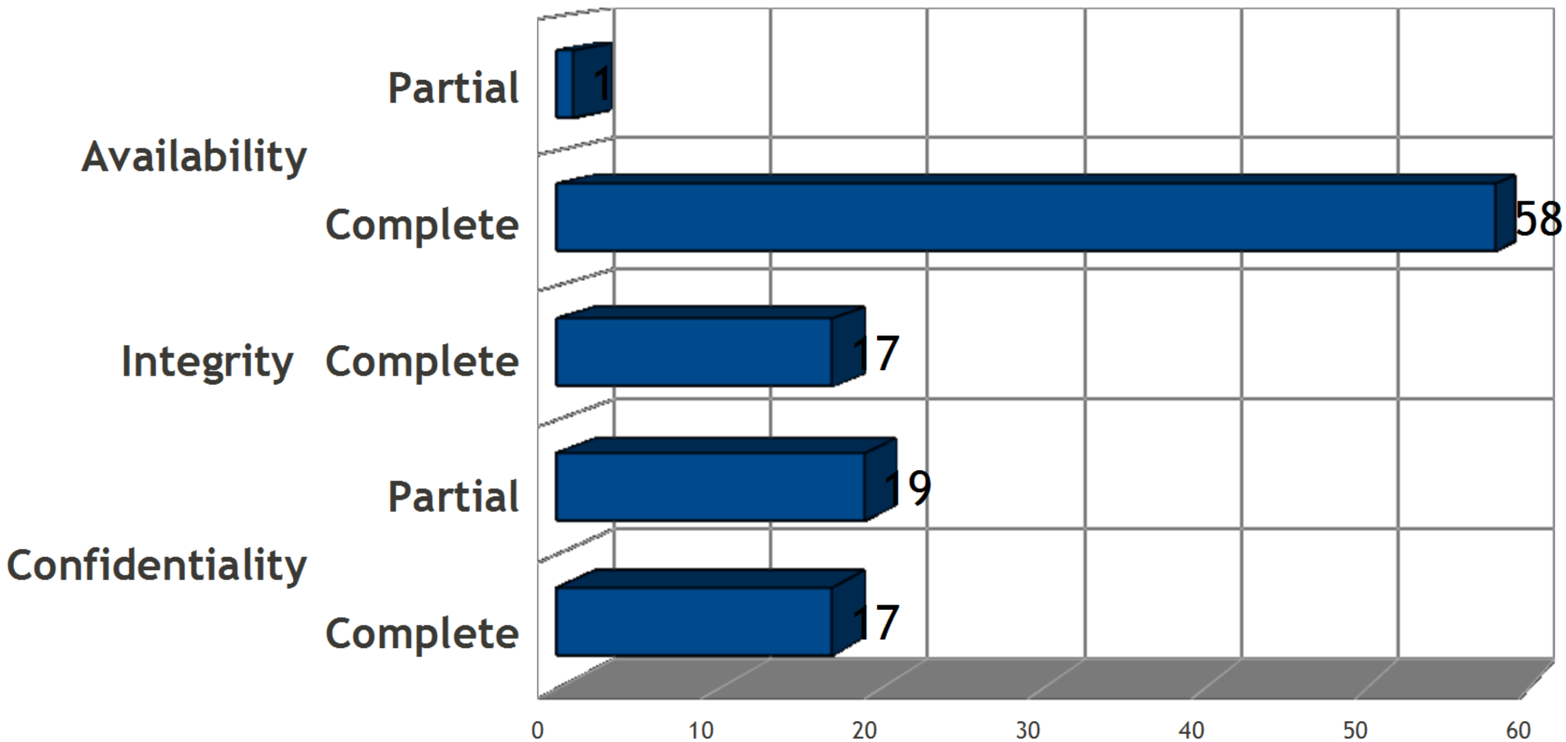
Analysis of 2011 Kernel Vulnerabilities Based on the Origin of the Attack



Analysis of 2011 Kernel Vulnerabilities Based on Access Complexity



Analysis of the 2011 Data Based on the Vulnerability Impact



Conclusions

- Kernel space tracing is better than user space tracing in detecting normal and anomalous behaviour
- Classification algorithms when classifying normal and abnormal software behaviour yield similar results
- Generalization of system calls can reduce false positive rates significantly
- Using n-gram representation of function calls reduce the training time of HMM by 31.96% to 48.44%
- Most of the Linux vulnerabilities are exploited through host based attacks

Future Work

- Experiment with trace abstraction techniques to further reduce the trace size and training time
- Study other anomaly detection mechanisms based on continuous monitoring of system usage
- Incremental analysis of host-based systems to multiple system processes
- Investigate additional generalization methods in detection of anomalies
- Investigate feedback-directed and self-adaptive anomaly detection techniques

Thank you!

SBA Research Lab: Contact Information

Dr. Wahab Hamou-Lhadj
Associate Professor

Mailing Address:
Department of ECE
Concordia University
1455 de Maisonneuve West
Montreal, Quebec H3G 1M8 Canada

Civic Address:
Department of ECE
Concordia University
1515 St. Catherine, West
Montreal, Quebec H3G 2W1 Canada

Tel: +1 514 848 2424 x.7949
Fax: +1 514 848 2802
Email: abdelw@ece.concordia.ca

References

- A. Valdes and K. Skinner, "Adaptive, Model-Based Monitoring for Cyber Attack Detection," in Proc. of third Intl. Workshop on Recent Advances in Intrusion Detection, LNCS, Toulouse, France, Oct. 2000, pp. 80-92.
- C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," in Proc. of 1999 IEEE Symposium on Security and Privacy, Oakland, USA, May 1999, pp. 133-145.
- D. Y. Yeung and Y Ding., "Host-based intrusion detection using dynamic and static behavioral models," Pattern Recognition, vol. 36, no. 1, pp. 229-243, Jan. 2003.
- G. Jiang and C. Ungureanu, and K.i Yoshihira H. Chen, "Multi-resolution Abnormal Trace Detection Using Varied-length N-grams and Automata," in Proc. 2nd Intl. Conf. on Automatic Comp., Seattle, USA, June 2005, pp. 111-122.
- J. A. Jones, M. J. Harrold A. Orso, "Visualization of program-execution data for deployed software," in Proc. of the ACM symposium on Soft. Visualization, San Diego, USA, June 2003, pp. 67-76.
- N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection," IEEE Trans. on Computers, vol. 51, no. 7, pp. 810-820, July 2002.

References (2)

- S. A. Hofmeyr, S. Forrest, and and A. Somayaji, "Intrusion detection using sequences of system calls," J. Comput. Security, vol. 6, no. 3, pp. 151-180, Aug. 1998.
- S. S. Murtaza, M. Gittens, and Z., Madhavji, N. H. Li, "F007: Finding Rediscovered Faults from the Field using Function-level Failed Traces of Software in the Field," in Proc. of CASCON 2010, Toronto, Canada, Oct. 2010, pp. 57-71.
- W. Lee and S.J. Stolfo, "A framework for constructing features and models for intrusion detection systems.," ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, pp. 227-261, Nov. 2000.
- W. Wang, X. H. Guan, and X. L. Zhang, "Modeling program behaviors by hidden Markov models for intrusion detection," in Proc. of Intl. Conf. on Machine Learning and Cybernetics, Shanghai, China, Aug. 2004, pp. 2830-2835.
- X. D. Hoang, J. Hu, and and P. Bertok., "A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference," J. Netw. Comput. Appl, vol. 32, no. 6, pp. 1219-1228, Nov. 2009.

References (3)

- V. V. Phohaha, “The Springer Internet Security Dictionary,” Springer-Verlag, 2002.
- P. E. Proctor, “The Practical Intrusion Detection Handbook,” Prentice Hall PTR, NJ, USA, 2001.
- V. Chandola, A. Banerjee, V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys, vol. 41(3), article: 15, July 2009.
- S. Kumar, and E. H. Spafford, “A pattern matching model for misuse intrusion detection,” In Proceedings of the National Computer Security Conference, Baltimore, MD, 1994, pp. 11–21.
- D. E. Denning, “An Intrusion Detection Model,” IEEE Transactions on Software Engineering, SE, vol. 13(2), 1987, pp. 222-232.
- S. Forrest, P. D’haeseleer, and P. Helam, “An immunological approach to change detection: Algorithms, analysis and implications”. In Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, vol. 110, 1996.
- D. Endler, “Intrusion detection: applying machine learning to solaris audit data,” In Proceedings of the IEEE Annual Computer Security Application Conference, Society Press, 1998, pp. 268 – 279.
- Guofei Jiang, Haifeng Chen, Cristian Ungureanu and Kenji Yoshihara, “Trace analysis for fault detection for application server”, Handbook of Automatic Computing: Concepts, Infrastructures, and Applications, edited by S. Hariri, and P. Parashar, CRC Press, 2007.
- C. Warrender, S. Forrest, and B. Pearlmutter, “Detecting intrusions using system calls: Alternate data models,” In Proceedings of the IEEE ISRSP. IEEE Computer Society, 1999, pp. 133 – 145.
- J. Hu, Q. Dong, X. Yu, and H. H. Chen, “A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection,” IEEE Netw. vol. 23(1), 2009, pp. 42 – 47.
- Jiankun Hu, “Host-Based Anomaly Intrusion Detection”, Handbook of Information and Communication Security, Springer, 2010.
- S. Forrest, S. A. Hofmeyr, A. Somayaji. and T. A. Longstaff, “A sense of self for unix processes,” In Proceedings of the IEEE ISRSP, 1996, pp. 120 – 128.
- E. Eskin, “Anomaly detection over noisy data using learned probability distributions,” In Proceedings of the 17th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., 2000, pp. 255–262.
- E. Eskin, W. Lee, and S. Stolfo, “Modeling system call for intrusion detection using dynamic window sizes,” In Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX), 2001.

References (4)

- A. K. Ghosh, and A. Schwartzbard, “A study in using neural networks for anomaly and misuse detection,” In Proceedings of the 8th USENIX Security Symposium, 1999.
- N. Abouzakhar, A. Gani, G. Manson, M. Abutbel, and D. King, “Bayesian learning network approach to cybercrime detection,” In Proceedings of the 2003 Post Graduate Networking Conference, Liverpool, United Kingdom, 2003.
- W. Hu, Y. Liao, and V. R. Vemuri, “Robust anomaly detection using support vector machines,” In Proceedings of the International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., 2003, pp. 282–289.
- G. Stein, C. Bing, A. S. Wu, and K. A. Hua, “Decision tree classifies for network intrusion detection with GA-based feature selection,” in Proceedings of the 43rd Annual Southeast Regional Conference, Georgia, 2005, pp. 136 – 141.
- Q. Xu, W. Pei, and Q. Zhao, “An intrusion detection approach based on understandable neural network trees,” Journal of Electronics, 2007, pp. 574 – 579.
- R. C. Chen, K. F. Cheng, Y. H. Chen, C. F., Hsieh, “Using Rough Set and Support Vector Machine for Network Intrusion Detection System,” In proceedings of the First Asian Conference on Intelligent Information and Database Systems, 2009, pp. 465 – 470.
- L. R. Rabiner and B. H. Juang, “An introduction to hidden markov models,” IEEE ASSP Magazine, 1986.
- P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, “Class-based n-gram models of natural language”, Computational Linguistics, vol. 18, pp. 467–479, 1992.
- Gzip Official Website http://www.gzip.org/
- M. Desnoyers, and M. R. Degenais, “The LTTng tracer: A low impact performance and behavior monitor for GNU/Linux,” In Proceedings of Ottawa Linux Symposium, Ottawa, Canada, July 19 – 22, 2006.
- LTTng Official Website. http://lttng.org
- Weka Official Website http://www.cs.waikato.ac.nz/ml/weka/
- Leonard E. Baum, Ted Petrie, George Soules and Norman Weiss, “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains”, The Annals of Mathematical Statistics, vol. 41(1), February, 1970, pp. 164 – 171.
- J. Han, and M. Kamber, “Data Mining: Concepts and Techniques,” 2nd edition, San Francisco: Elsevier, 2006.
- J. Langford: Optimizing hidden Markov model learning, Technical Report (Toyota Technological Institute at Chicago, Chicago 2007)