Intrusion Detection and Forensics Using Series of Function Calls

Sean Peisert
UCSD

February 21, 2006

Talk for UCI Graduate "Language-Based Security" Class

University of California San Diego

How Do We Do “Good” Forensics & ID?

- Forensic tools help to analyze the data
  - Data helps understand what went wrong
  - Data is descriptive
- Real-time ID tools help to find the anomaly
  - Data is useful for automated “search”
- Why can’t we have both?

Possible System Abstractions?

- Assembly Code
- Syslog Messages
- Login/Logouts
- Resource Usage Metrics
- System Calls

Forensic Principles

- Consider the entire system
- Consider actions and their effects.
- Runtime data is the only authoritative record of what happened.
- Actions and results must be processed and presented in a way that is understandable by humans.

Intrusion Detection

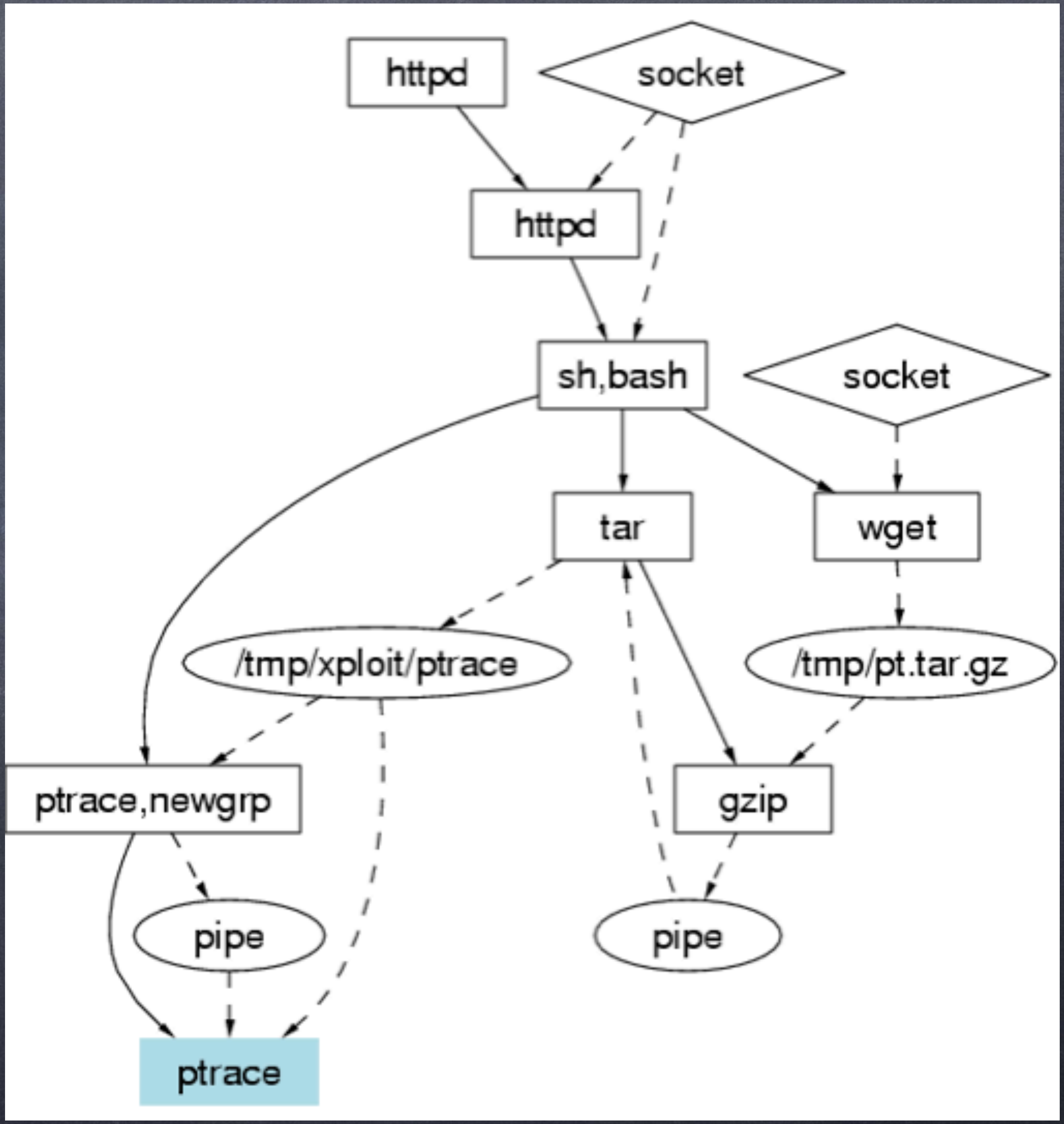
- Anomaly Detection
- Anomaly Detection w/Automated Rule Generation
- Signature (Misuse) Detection

Anomaly Detection

- D. Denning in 1986
- Immunological Approach (Forrest, et al.)
- Data Mining

Forensics

- syslog
- "Analyzing Computer Intrusions" (Andrew Gross)
- BackTracker (King & Chen)



Research Questions

- Can we improve post-hoc anomaly detection accuracy by using function calls as data, as opposed to system calls alone?
- Can we enable forensic analysis of "intrusions" not otherwise possible or easy?

Methods

- Capture all calls, their arguments, and their return values
- Compare series of calls between "safe" set and "test" set
- Future: Compare arguments and return values between "safe" set and "test" set

Hamming Distance

- Example:
  - "Safe": a b c d e f
  - "Test": a b c d e g
  - Hamming Distance (d) = ?
  - Min sequence length to find anomaly = ?

Minimum Hamming Distance

- "Safe" Corpus:
  - Size 2: e f, f c, f a, f b,
  - Size 3: e f a, e f b, f f c
- "Test" Sequence
  - e f c
- What is the minimum sequence length required to detect this as an anomaly?

Immunological Approach

- Sliding window of size k
- "Safe" sequences j
- "Test" sequences i
- d_{min} = min{d(i,j) for all safe sequences j}
- $\hat{S}_A$ = max{d_{min}(i) for all new sequences i}/k

Analyzing Function Arguments & Return Values

- Can't use the same techniques — need more advanced data mining
- Clustering: k-nearest-neighbor, k-means

Forensic Methods

- Prefer to have source code to search for captured calls

Gathering Data

- Variety of methods:
  - Virtual Machine (a la "Introvirt")
  - Binary Rewriter/Dynamic Instrumentation
  - Compiler
  - Intel's Pin (Luk & Cohn, et al., PLDI 2005)

su Experiment #1

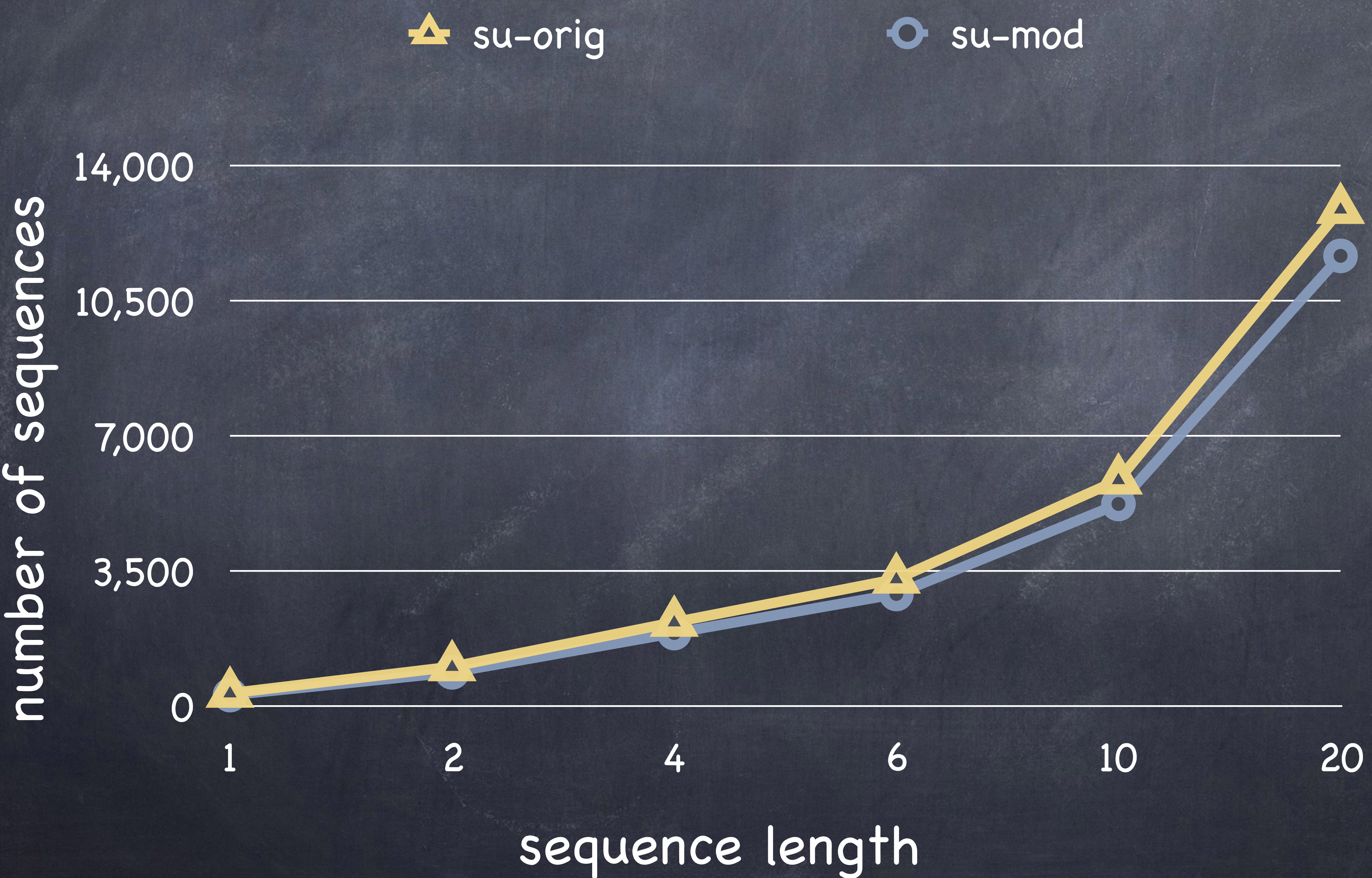
- Removed call to pam_authenticate(). What changed?

k=4	total calls	total seq	unique seq
su-orig	88208	51085	2170
su-mod	49453	30669	1891

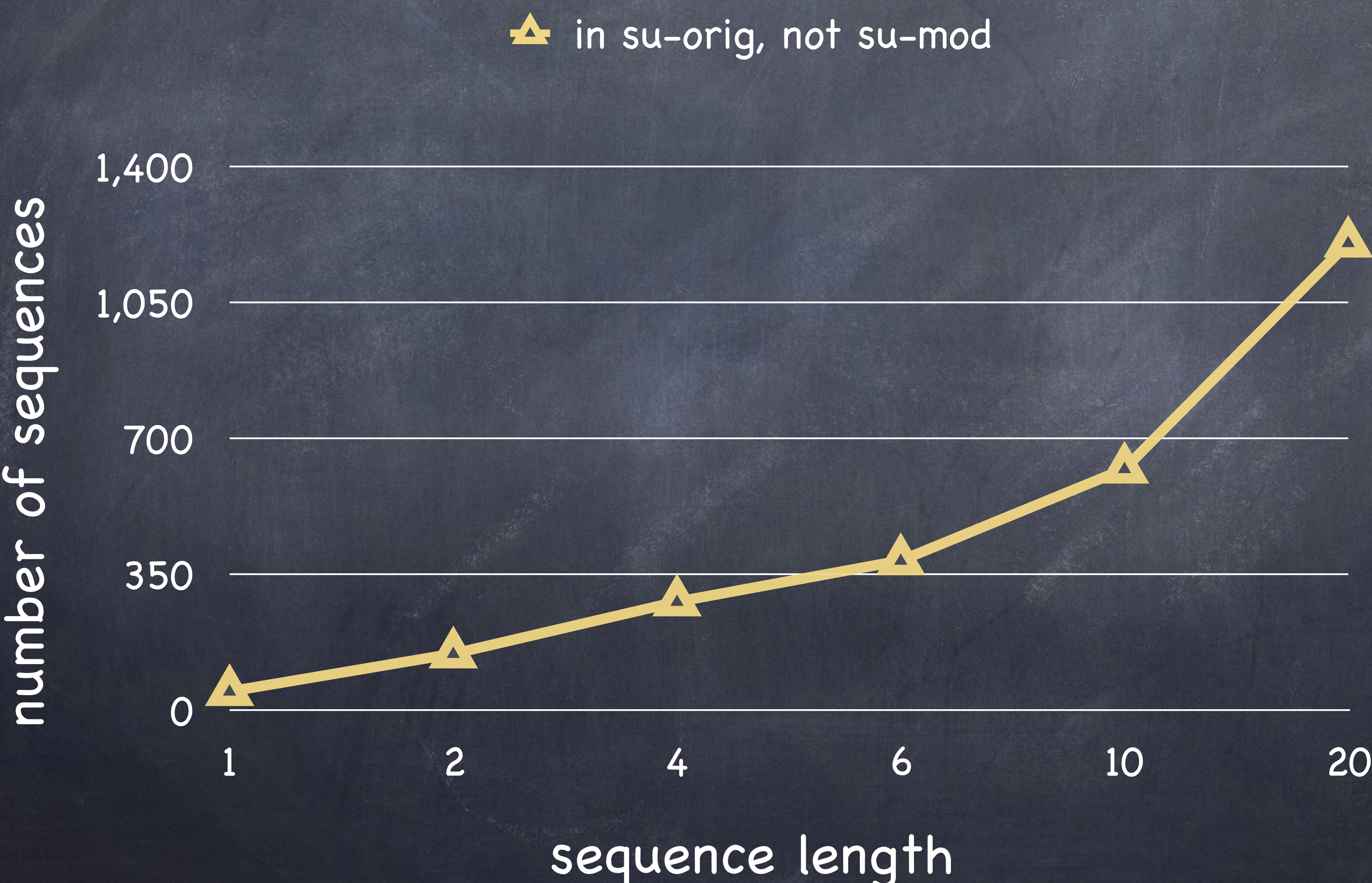
su-orig vs. su-mod

k=4	total different seqs
only in su-orig	18453 (315 unique)
only in su-mod	36 (all 36 unique)

Total Function Call Seqs in su



Difference in Total Function Call Seqs in su



Calls in su-orig not in su-mod

sequence	# total occurrences	% of total program
MD5Update	5538	10.85%
MD5Final	2005	3.92%
MD5Init	1002	1.96%
MD5Pad	1002	1.96%
Total	9547	18.69%

su Experiment #2

- Ignored result of pam_authenticate() call

k=2	total different seqs
only in su-orig	2594 (2379 unique)
only in su-mod	2 (both unique)

One of 2 seqs: strcmp , pam_authenticate

ssh Experiment #1

- Edited ssh to echo the password to the terminal

k=4	total different seqs
only in ssh-orig	12 (9 unique)
only in ssh-mod	47 (38 unique)

vfprintf, vfprintf, fprintf, read_passphrase

ssh Experiment #2

- Edited ssh to send the password through a network socket

k=4	total different seqs
only in ssh-orig	14 (14 unique)
only in ssh-mod	45 (45 unique)

inet_aton, inet_addr, rtld_free_tls, rtlf_free_tls

lpr Experiment

- Recreated UNM experiment that exploits lpr bug.
- Exploits counter, “creat” syscall, and symlink to rewrite /etc/passwd.

lpr Results

only in lpr-orig	only in lpr-mod
seteuid, error_unthreaded	sys_write, close
sbrk, sys_umask	lseek, sys_write
open, sys_umask	copy, close
	nfile, sys_read
	creat, sys_umask
	sys_read, sys_write
	sys_read, sys_syscall
	open, creat
	sys_unlink, error_unthreaded
	close, copy
	close, close
	close, seteuid
	sys_umask, fchown

Conclusions

- These initial experiments seem to help highlight anomalies and then help understand them.

(Immediate) Future Work

- More experiments (including blind and/or double-blind ones)
- Arguments & return values
- Machine learning applied to function calls
- Tuning parameters

Key References

- Bace, “Intrusion Detection,” 2000.
- Denning, “An Intrusion-Detection Model,” IEEE TSE, 1987.
- Gross, “Analyzing Computer Intrusions,” UCSD PhD Thesis, 1997.
- Hofmeyr, Forrest, & Somayaji, “Intrusion Detection Using Sequences of System Calls,” JCS, 1999.
- Lee & Stolfo, “Data Mining Approaches for Intrusion Detection,” USENIX Security, 1998.
- Peisert, Bishop, Karin, & Marzullo, “Principles-Driven Forensic Analysis,” NSPW, 2005.
- Warrender, Forrest, et al., “Detecting Intrusions Using System Calls: Alternative Data Models,” IEEE S&P, 1999.

Acknowledgements

- This work was partially funded by a Lockheed-Martin Information Assurance Technology Focus Group 2005 University Grant, and NSF.