BunnyTN3
Terzo Workshop di Crittografia
Trento – March 12, 2012

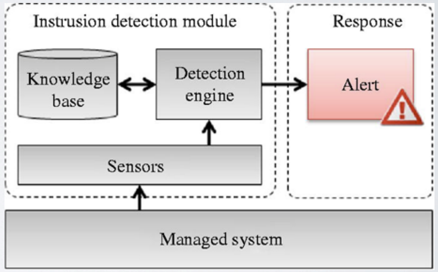
UNOBSERVABLE INTRUSION DETECTION BASED ON CALL TRACES IN PARAVIRTUALIZED SYSTEMS

Marino Miculan
University of Udine
Google: miculan

(Work in collaboration with Carlo Maiero)

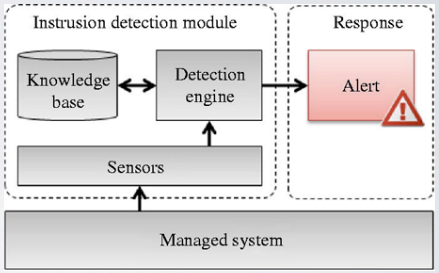
Intrusion Detection Systems

- IDS gathers data from the management system (via "sensors") and using a KB decides if to raise an alert
- Crucial design questions:
  - **What** to observe?
  - **How** to observe?



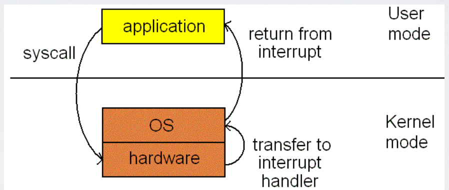
What To Observe?

- "Syntactic" IDS look for discrepancies in code, data… (virus signatures, digests of programs, patterns…)
- Quite limited
  - Patterns change often
  - (Antivirus detect ~50% viruses)
  - Difficult to look into process memory (e.g. to detect buffer overflows)



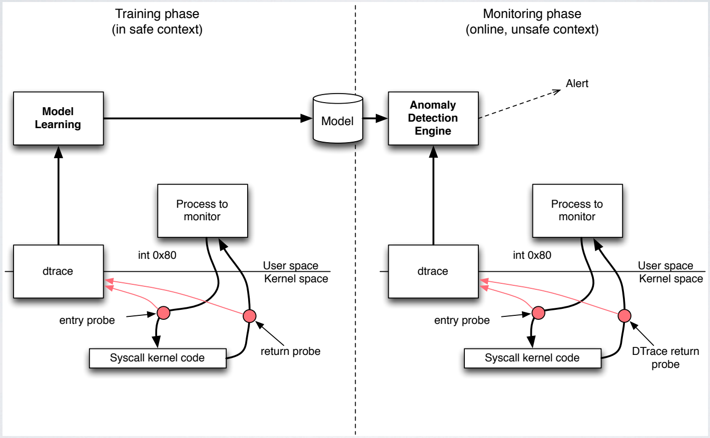
What To Observe?

- "Semantic" IDS: look for discrepancies in the run-time ***behavior*** with respect to the expected one (the "model")
  - More robust to changes, non intrusive, …
- Behavior = interactions with environment



- **Slogan: A process behavior is fully determined by its system call traces** (with parameters)
- Black box approach: no need to look "inside" the application

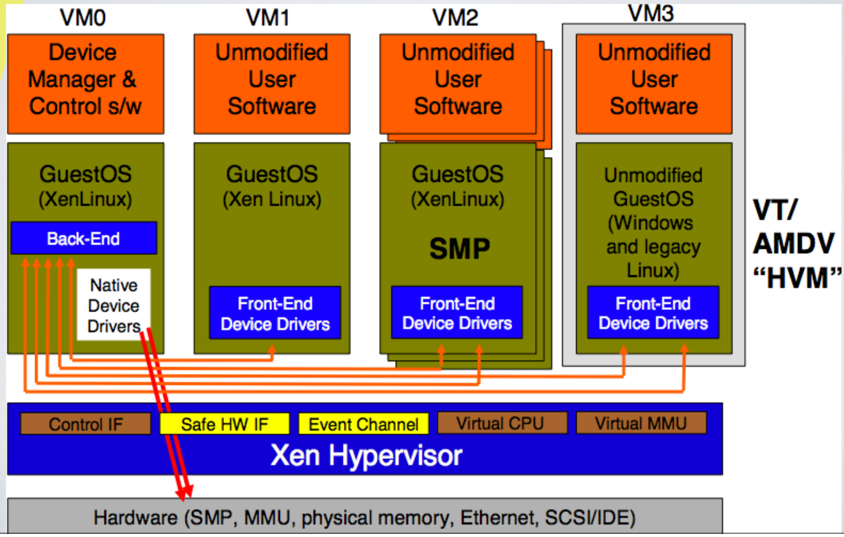
First (naive) Architecture



But The Enemy Is Smart…

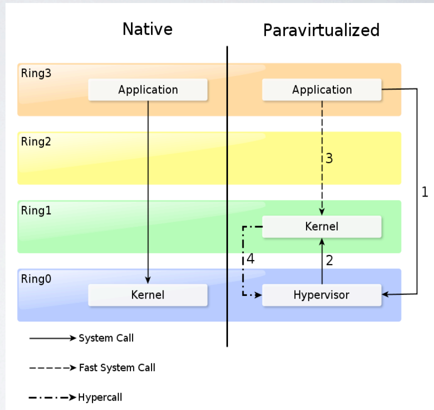
- First architecture requires changes in Operating System kernel in order to place probes on system calls
- Attacker can
  - **notice** the presence of probe, and change attack accordingly
  - **attack** the IDS itself, by removing probes
- How to observe system call traces WITHOUT changing OS?

Solution: Paravirtualized Systems



How To Intercept Syscall In VM

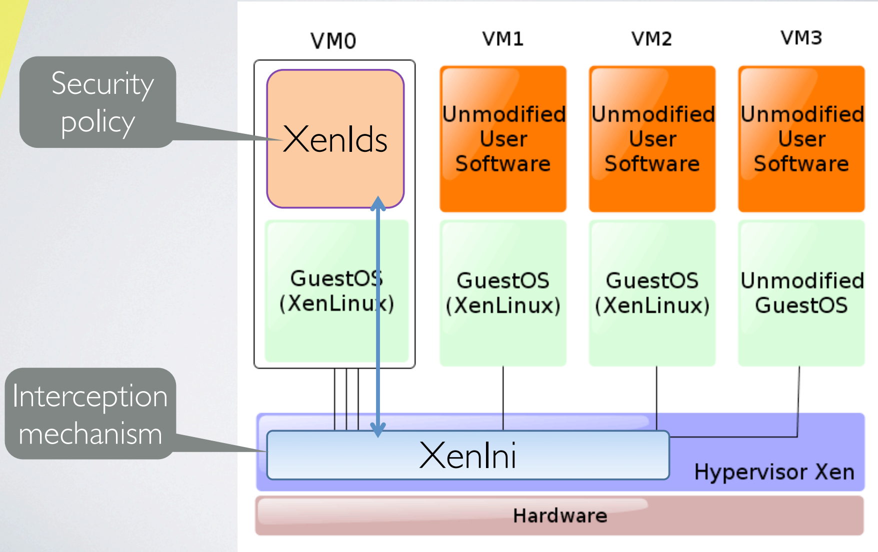
- In paravirtualized system system calls are trapped in a different way
- What and where to intercept?



0x80	**EAX**	EBX	ECX	EDX	EBP	ESI	EDI	ESP	**EIP**

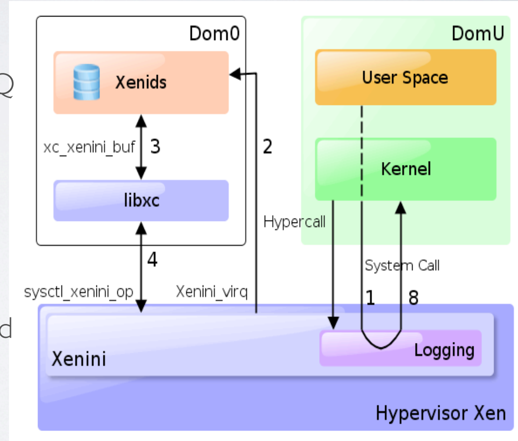
System call Number | Pid

New Architecture: XenIDS



Stealth Interception

1. Xenini intercepts the system call or the hypercall
2. Xenini alerts XenIds via a VIRQ
3. XenIds makes a request get info to libxc
4. Libxc requires data to Xenini
5. Xenini transmits the data to libxc
6. Libxc returns data to IDS
7. the IDS processes the data and gives an answer.
8. control flow returns to the guest VM

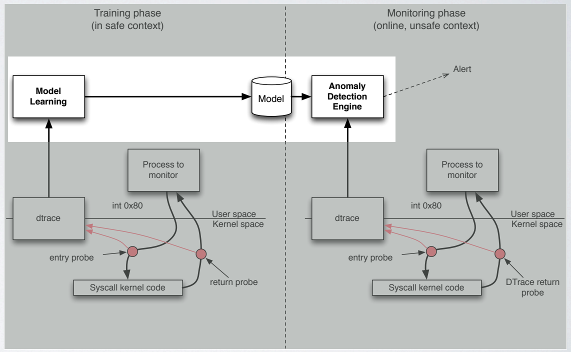


Advantages Of XenINI/XenIDS Architecture

- **Secure**: does not change any guest kernel structure, thus cannot be tampered
- **Isolated and unobservable**: the attacker cannot tell whether is monitored or not
- **Flexible** and independent from virtual machine
- **Independent from memory**: no introspection in guest memory or disk
- **Simple**: only one point of deployment

Model Construction And Anomaly Detection

- So we can observe system call traces without being catch
- What should we do with these traces?
- Various methods to construct model & detect anomalies

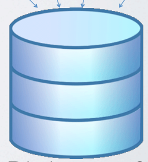


- We will see only a simple one (we are working also on others)

Algorithms For Anomaly Detection: Stide

- Stide looks for *suspect* subsequences of syscalls
- **Model**: All subsequences of lenght *k* of normal execution (patterns) of all programs running on a machine (usually k=5 or 6)
- **Learning:** All pattern generated by a machine during normal execution are stored in database
  - This can lead to more false negative in a server running many programs, but not more false positives. (Not observed in our tests)

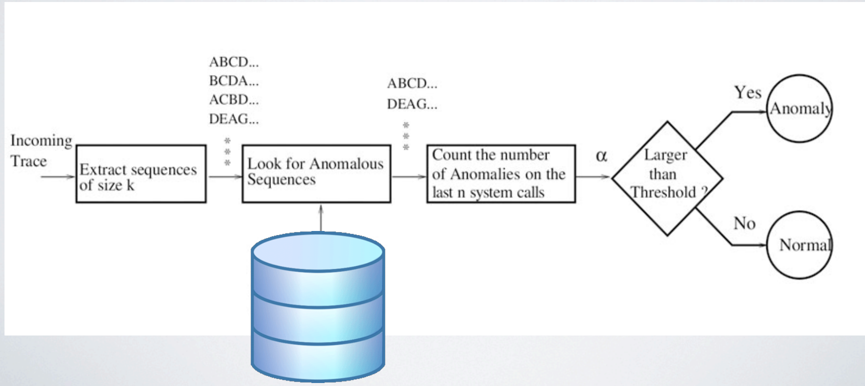
P1	P2	P3	P4
5	21	2	2
3	5	5	5
4	3	2	4
3	4	3	3
4	22	1	6



Dictionary of normal sequences

Algorithms For Anomaly Detection: Stide (cont.)

- **Detection**: an intrusion is recognized only if the number of anomalies on the last *n* syscalls is > threshold.



About The Threshold

- Choosing the threshold Th is crucial
  - Low Th => too many false positives
  - High Th => attacks with less anomalies than Th are not detected (false negative)
- For our test, after two weeks of training period we identified Th as 15%
  - No false positives
  - Behaviors differing less than 15% from stored sequences are considered "safe"

Stide: Evaluation Of Detection Capability

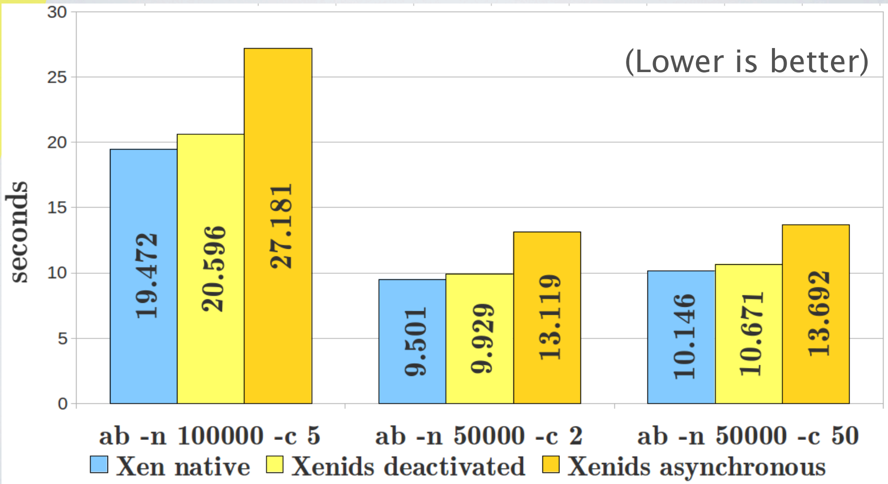
- Offline test on M.I.T. interception traces: all attacks have been recognized, no false positives
- Online test: observation of a modified (i.e. "hacked") FTP server

Change to FTP server	Mismatch	Anomaly?
local copy string	20%	Yes
open a system shell	50%	Yes
remote copy string	30%	Yes

- Observation of normal uses which did not appeared in training set

Use	Mismatch	Anomaly?
strings of 25 chars	< 15%	No
strings of 100 chars	< 15%	No
closing using kill	< 15%	No

Stide: Performance Evaluation



Overall overhead: 7–8% (in asyncronous mode)

Conclusions

- We have shown how to detect host intrusions by observing only system calls, without being observed by the intruder
- The overhead of XenIDS is acceptable for real time detection
- Threshold is delicate: it depends on various aspects
  - the training period
  - the desired "aggressiveness" of the IDS
- To circumvent these issues, we are working on new models based on *Execution Graphs* extended with *Data Flow constraints*

Thanks For Attention

Questions?

marino.miculan@uniud.it