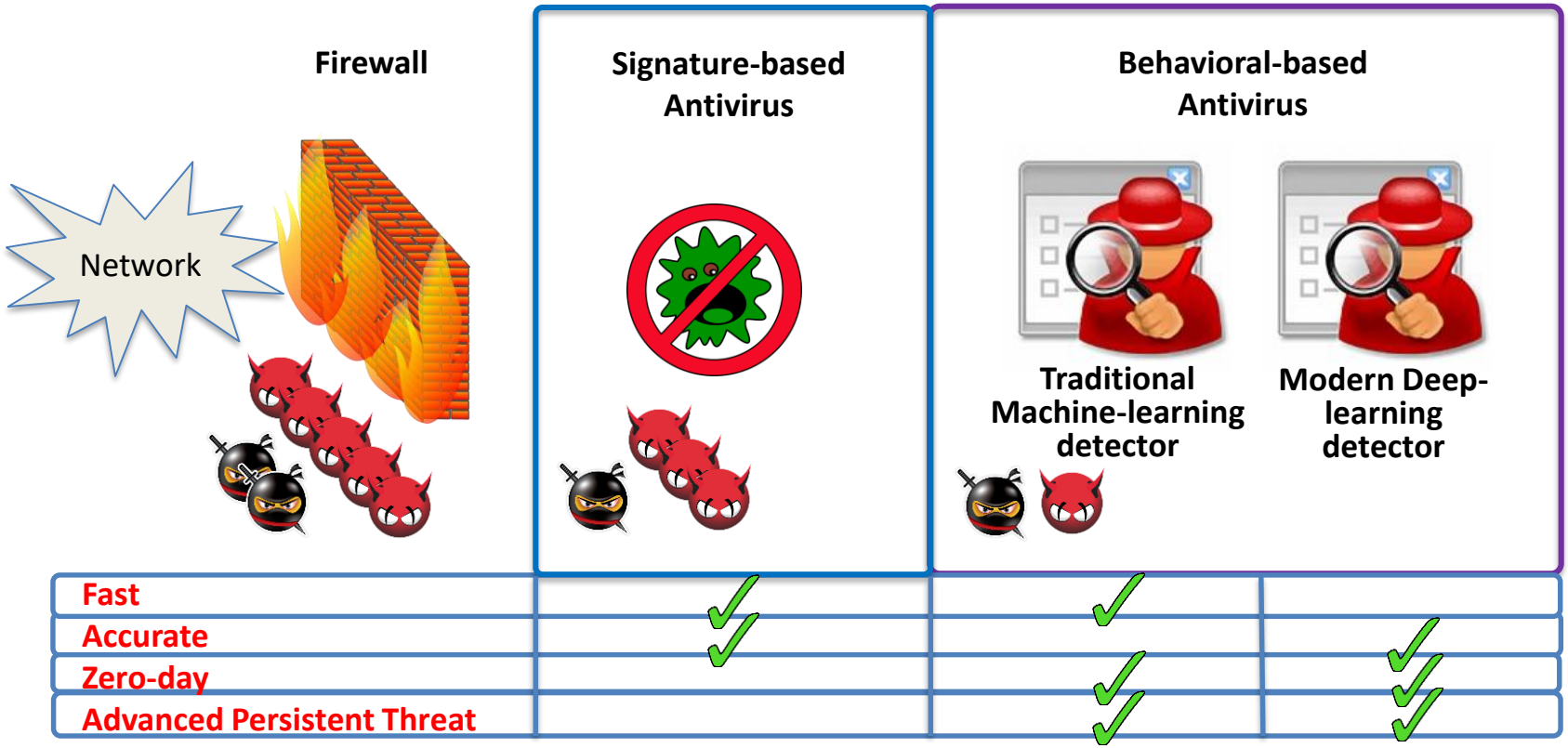
The Dose Makes the Poison - Leveraging Uncertainty for Effective Malware Detection

Ruimin Sun,[1] Xiaoyong Yuan[1], Andrew Lee[2], Matt Bishop[3], Donald E. Porter[4], Xiaolin Li[1], André Grégio[5], Daniela Oliveira[1]

[1]University of Florida(US), [4]University of North Carolina at Chapel Hill(US), [3]University of California at Davis(US), [5]Federal University of Parana(Brazil), [2]Duke University(US)

Malware Detection

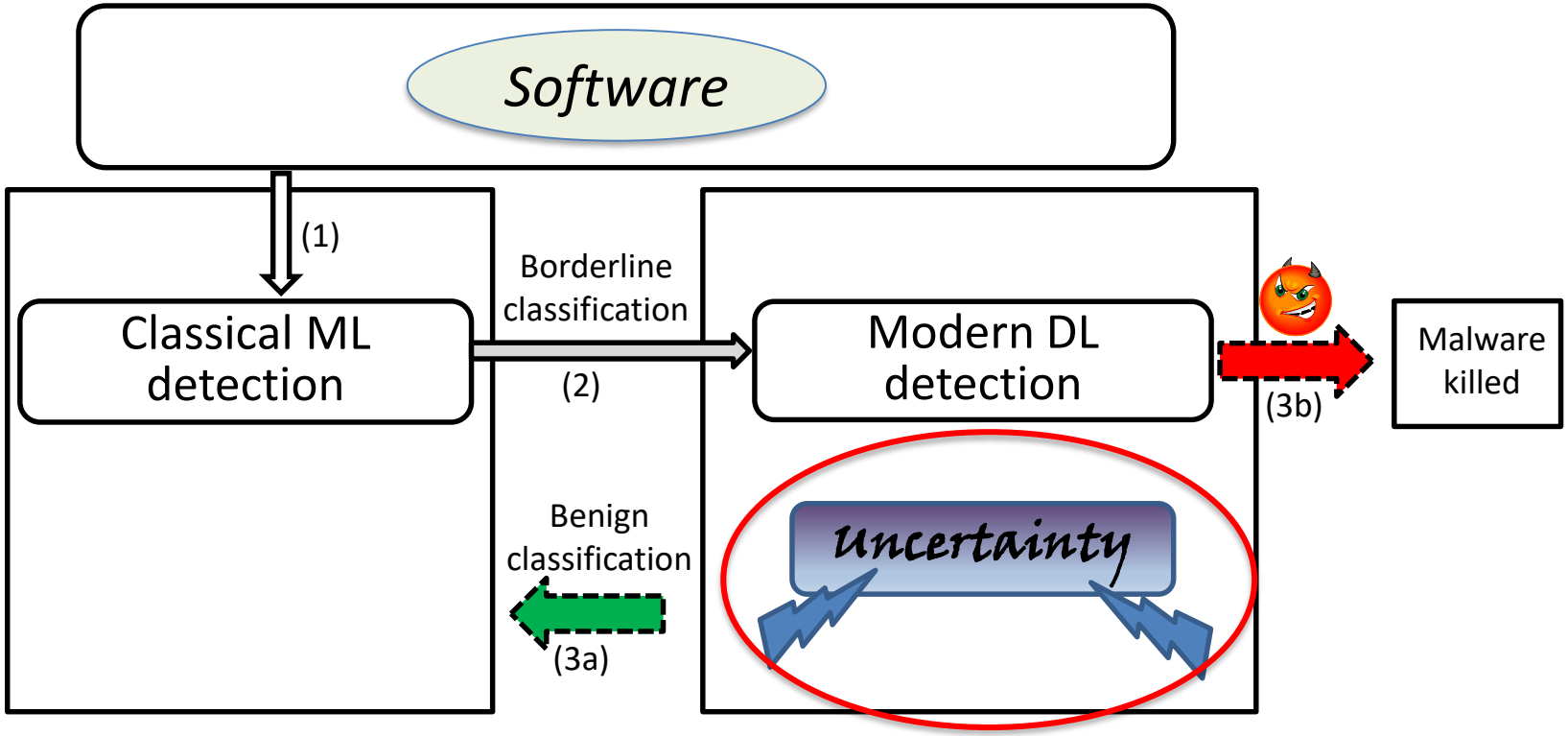


Motivation

- Resourceful attacker can eventually get in.
- Pure traditional ML and DL has Pros and Cons .
  - Why not combine the best of the two worlds?
- Rate-limit potential malware is in need.

Chameleon

- Making the combination of ML and DL possible



Interference Set

- Interference Set
  - 37 **system calls** representing OS functionalities relevant for malware
  - Most are I/O-bound

Category	System call
File related	sys_open, sys_openat, sys_creat, sys_read, sys_readv, sys_write, sys_writev, sys_lseek, sys_close, sys_stat, sys_lstat, sys_fstat, sys_stat64, sys_lstat64, sys_fstat64, sys_dup, sys_dup2, sys_dup3, sys_unlink, sys_rename
Network related	sys_bind, sys_listen, sys_connect, sys_accept, sys_accept4, sys_sendto, sys_recvfrom, sys_sendmsg, sys_recvmsg, sys_socketcall
Process related	sys_preadv, sys_pread64, sys_pwritev, sys_pwrite64, sys_fork, sys_clone, sys_nanosleep

Strategies

- Perturbations to software
  - E.g. slow down, temporary function lost
  - *Non-intrusive* Strategies for whitelisted software
    - System call silencing with error return
    - Process delay
    - Process priority decrease
  - *Intrusive* strategies for non-whitelisted software
    - System call silencing
    - Buffer bytes change
    - Connection restriction
    - File offset change

Chameleon Architecture

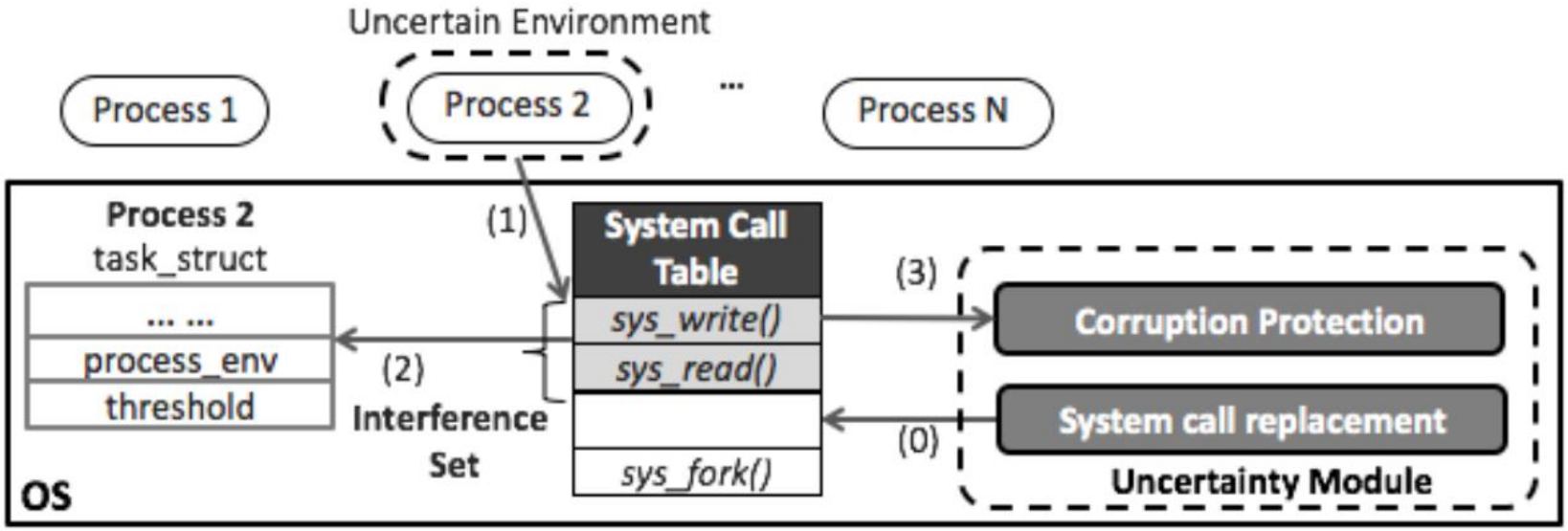


Fig. 1: **System architecture**. When a process running in the uncertain environment invokes a system call in the interference set (1), the ***Uncertainty Module*** checks if the process is running in the uncertain environment (2), and depending on the execution of the ***corruption protection*** mechanism (3), **randomly** selects an interference strategy to apply to the system call. The corruption protection mechanism prevents interferences during accesses to critical files, such as libraries.

Experiments

- 113 software
  - From GNU projects, SPEC CPU2006, and Phoronix-test-suite
  - 47 I/O-bound and 66 CPU-bound
- 100 Linux malware
  - From THC and VirusShare
  - 22 flooders, 14 worms, 15 spyware, 24 Trojans and 25 viruses
- Threshold
  - 10%, 50%
- Logging execution-related data
  - whether or not the program was adversely affected
    - *Succeeded, Hampered, Crashed*
  - the number of invoked system calls

Evaluation Results

	Threshold = 50%		Threshold = 10%	
Malware Category	Intrusive	Non-intrusive	Intrusive	Non-intrusive
Spyware	27%	40%	53%	60%
Viruses	24%	24%	24%	28%
Worm	21%	21%	29%	21%
Trojans	17%	29%	46%	38%
Flooders	9%	9%	41%	18%
All	**19%**	**24%**	**38%**	**32%**

	Threshold = 50%		Threshold = 10%	
Software Category	Intrusive	Non-intrusive	Intrusive	Non-intrusive
Text Editors	0%	33%	53%	73%
Compilers	18%	36%	55%	73%
Network Tools	38%	50%	56%	56%
Scientific Tools	33%	40%	53%	60%
Others	82%	79%	86%	86%
All	**41%**	**51%**	**63%**	**70%**

The ratio of **Succeeded** execution.

Evaluation Results

Malware Category	# of syscalls monitored	% of syscalls perturbed	% of connection-related syscalls perturbed	% of buffer-related syscalls perturbed
Spyware	50.37	2.89%	7.14%	3.06%
Viruses	423.44	5.02%	9.56%	4.96%
Worm	68880.64	0.05%	9.86%	8.97%
Trojans	523.80	8.09%	9.52%	7.14%
Flooders	930.50	9.74%	10.13%	6.58%
All	**9992.49**	**0.41%**	**9.87%**	**6.83%**

Goodware Category	# of syscalls monitored	% of syscalls perturbed	% of connection-related syscalls perturbed	% of buffer-related syscalls perturbed
Text Editors	6693.20	0.42%	0.04%	0.40%
Compilers	167303.36	0.04%	0.00%	0.00%
Network Tools	515.50	2.85%	10.99%	1.54%
Scientific Tools	2071.59	1.13%	0.00%	0.46%
Others	566.31	0.54%	0.00%	0.19%
All	**20863.74**	**0.10%**	**0.40%**	**0.03%**

Comparison on ***system call*** perturbation (with Non-intrusive strategies at threshold 10%).

APT

- Simulated watering hole attack similar to *the **Black Vine** **APT*** from Symantec

Environment	None	Threshold = 10%		Threshold = 50%	
		Non-intrusive	Intrusive	Non-intrusive	Intrusive
# of syscalls monitored	85	81	82	20	25
% of syscalls lost	0	5	4	76	71
% of connection-related syscalls lost	0	8	6	79	69
% of buffer-related bytes lost	0	9	9	95	96

Execution details of the **APT** in the standard and uncertain environment.

Discussion

- A resourceful adversary can bypass *any* protection mechanism
  - Highly fault-tolerant malware can escape the uncertain OS.
- There are *trade-offs* in selecting an interference strategy
  - Intrusive strategies are more aggressive.
  - Suitable for organizations with high security demands.
- The *worst* case scenario for effectiveness
  - SW receiving borderline classification all the time.
    - Stays in the uncertain environment.
- Can be adapted to *Windows* as well

Conclusions

- CHAMELEON
  - A Linux framework using uncertainty to rate-limit possible malware.
  - Provides a “safety net” for failures of standard intrusion detection.
- Results
  - Malware were ***disproportionately disrupted*** by the uncertain environment than common software (38% vs. 70%).
- Other contributions
  - Making systems diverse by design.
  - Increasing attackers’ work factor.
  - Decreasing the success probability and speed of attacks.
  - Supports the combination of traditional ML and emerging DL methods.

Thank you!

Questions?

Ruimin Sun
gracesrm@ufl.edu