FAST AND SERVICE-PRESERVING RECOVERY FROM MALWARE INFECTIONS USING CRIU

ASHTON WEBSTER
RYAN ECKENROD
DR. JAMES PURTILO

INTRODUCTION

- Malware infections are essentially inevitable at scale
- Most malware removal tools are excellent at undoing malware changes
- ...but what about availability of system?

MOTIVATING EXAMPLE

- Running a web server
- Periodically, web server is infected by malware and must be restarted
- Is there a better way to preserve active (benign) connections and processing state through the restore?

STATE OF THE ART: THE NAIVE APPROACH

- “Turn it off and on again” (and reformat drives)
- VM Snapshots
- Antivirus Restore

STATE OF THE ART: LOG BASED AND VM BASED METHODS

Project/Name	Space Required	Runtime Overhead	Restore Overhead	Reverts all “bad” state?	Recovers all “good” state?	Maintains active connections?
Taser (Goel et al.)	On the order of GBs per day for logs	~7%	Minutes to hours	In virtually all cases	In virtually all cases	No

DESIGN AND IMPLEMENTATION

OUR SOLUTION: CRIU-MR

- Leverages existing technologies LXC and CRIU
- Preserves active connections
- Recovery process takes seconds
- Virtually no overhead during runtime
- Malicious process state saved for forensic analysis

SOLUTION COMPONENTS: LXC – LINUX CONTAINERS

- **Virtualization and Sandboxing for Linux using containers**
- **Come in privileged and unprivileged varieties**
  - Privileged Containers run as root and are not considered secure
  - Unprivileged containers run as an unprivileged user and map uids and guids to random ranges on the actual syste

SOLUTION COMPONENTS: CRIU - CHECKPOINT AND RESTORE IN USERSPACE

- **Saves state of individual Linux processes in image files**
- **Able to restore TCP connections using TCP_REPAIR socket option**
    - Araujo et al. use this TCP restore functionality to dynamically restore infected containers to honeypots
- **Able to checkpoint and restore entire Linux containers as well**

HOW WE DID IT: CRIU-MR OVERVIEW

- **Modify CRIU for Malware Recovery**
  - During checkpoint, identify malicious processes/files/connections matching policies
  - During restore, omit processes identified during checkpoint
  - No changes needed for restoring legitimate connections
- **Create Agent for receiving alerts from IDS/IPS/etc.**
  - Create policies which can be read by our system to identify malware processes and modified state

CRIU-MR POLICIES

Created policy language flexible enough to handle variety of alerts

- **Static policies**
  - Assertions about state of container that should always hold
  - Stored as static input during startup of CRIU-MR agent
  - Example: Some process should never have a child process
- **Dynamic policies**
  - Additional information gathered by external IDS/IPS/AV scanner used to identify malware
  - Sent as JSON alert to CRIU-MR agent and dynamically included in policy

POLICY MATCHES

- **Executable Name Match**
- **Filename Match**
- **TCP IP Match**
- **Memory Match**
- **PID Match**
- **Parent PID Match**
- **Parent Executable Name Match**

IMPLEMENTATION: CRIU MODIFICATIONS

- **Total of 659 lines of C code added to fork of open source CRIU repository[1]**
- **Checkpoint**
  - Reads protobuf formatted policy file
  - Hook into resource serialization to check for policy elements
  - Write violating process IDs to file `omit.img`
  - Malicious process image information is saved
- **Restore**
  - Read back `omit.img`
  - At point of restore for each process, check if it is in omitted list
  - Don't restore processes with missing state (i.e. missing files)

https://github.com/ashtonwebster/criu

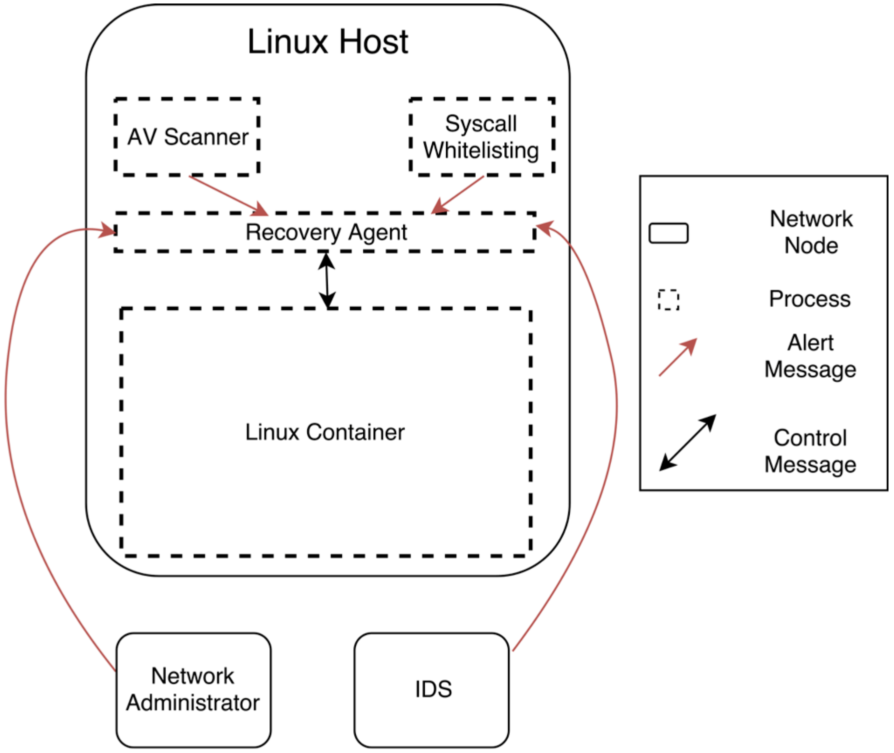
IMPLEMENTATION: LXC

- `lxc` - command to manage containers
- Checkpointing via CRIU
- Trivial changes to allow for added modified CRIU version
- Open source fork of the original repository[1]

[1]https://github.com/ashtonwebster/lxc

IMPLEMENTATION: CRIU-MR AGENT

- Simple python script to interface with modified CRIU/LXC
- Accepts JSON alerts and creates policies
- Orchestrates checkpoint, filesystem recovery, and restore
- Available as github repository[1]



[1] https://github.com/ashtonwebster/CRIU-MR-agent

CRIU-MR AGENT: FILESYSTEM RESTORE

- Assume that filesystem is “mostly static”
- Keep copies of container filesystem on host
- Quickly replace using *mv* command

INFECTION RECOVERY STEPS

1) Infection - Malware is introduced to the system
2) Detection - An AV Scanner, IDS, IPS, or other alert is generated and sent to the CRIU-MR agent as JSON alert
3) Preparation - JSON alert is transformed into a protobuf formatted policy, which is in turn passed to our modified version of CRIU
4) CRIU Checkpoint - all images generated; processes in violation of policy written to omit.img
5) Filesystem Restore - The backup system is placed at the container root location and the infected filesystem is moved to a different location
6) CRIU Restore - Non-malware processes are restored

EXPERIMENTS

EXPERIMENT I: MALWARE RECOVERY TIME

How long does it take to remove malware?

Experiment Outline:

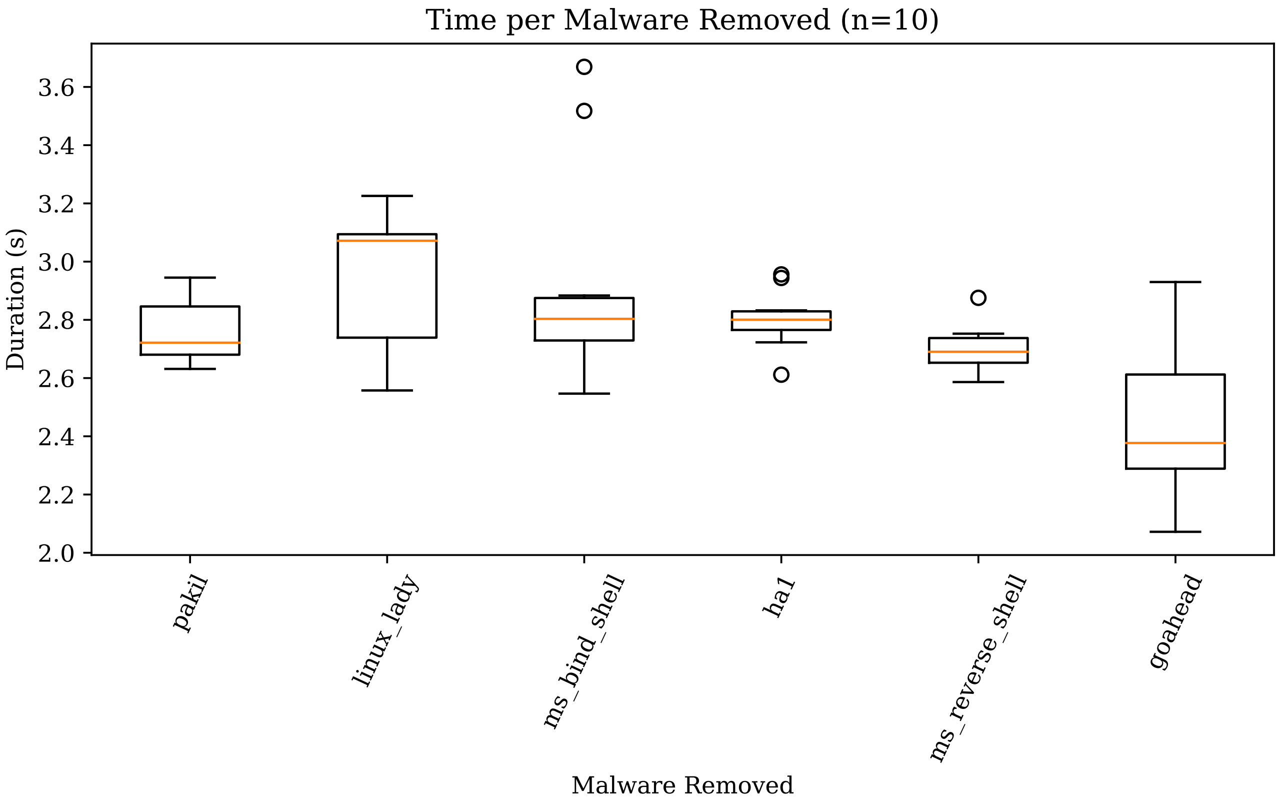
1) Initial clean state of container started
2) Malware started as root in background on container and allowed to run for 3 seconds
3) Detection is triggered and recovery starts

We repeat this removal process 10 times for each of 6 malware

EXPERIMENT I: MALWARE SELECTION

- *linux_lady* : Malware attempting to mine bitcoin via cronjob
- *ms_bind_shelll* : Metasploit exploit which binds on a port and provides a shell
- *ms_reverse_shell* : Metasploit exploit which starts a reverse shell from port
- *wipefs* : bitcoin mining executable
- *Linux.Agent* : Attempts to exfiltrate */etc/shadow* or */etc/passwd*
- *goahead_ldpreload* : An exploit on the GoAhead embedded webserver

EXPERIMENT I: MALWARE RECOVERY TIME RESULTS



EXPERIMENT I: MEAN (STD. DEV.) DURATION PER STEP

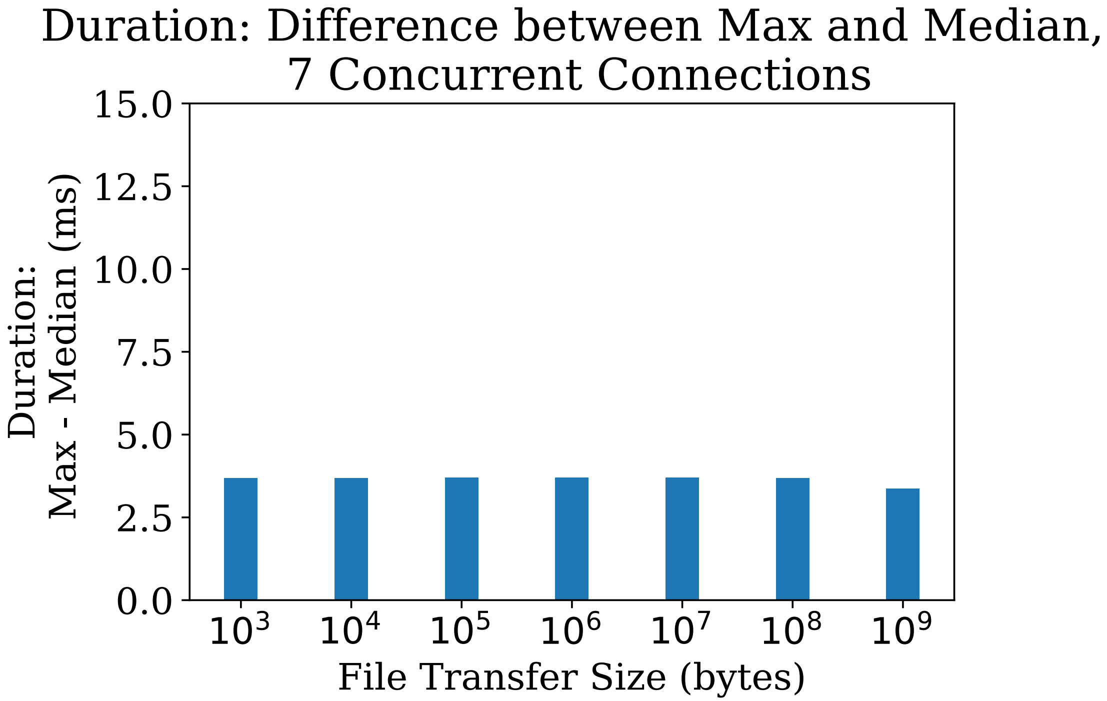
Step	Duration (s)
Preparation	0.02 (0.01)
Checkpoint	2.16 (0.20)
Filesystem Swap	0.01 (0.01)
Restore	0.57 (0.11)
Total	2.67 (0.27)

EXPERIMENT II: AVAILABILITY IMPACT STRESS TEST

What is the availability impact of recovering from malware?

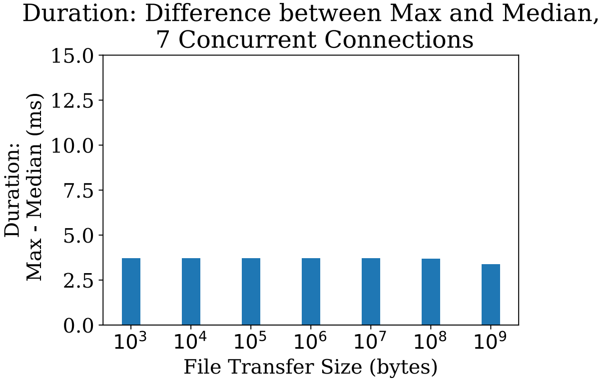
- 7 file sizes ranging from 1KB to 1GB by powers of 10 requested concurrently
- Experiment lasts for 1 minute
- At 30 seconds, malware is triggered, runs for 3 seconds, and recovery is triggered
- Time for each request is recorded
- In all cases, we find that no connections were terminated

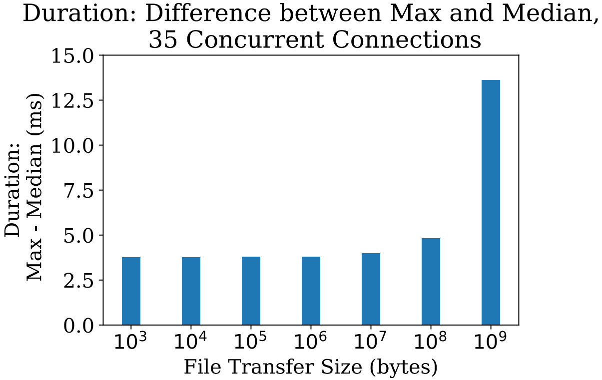
EXPERIMENT II: RESULTS

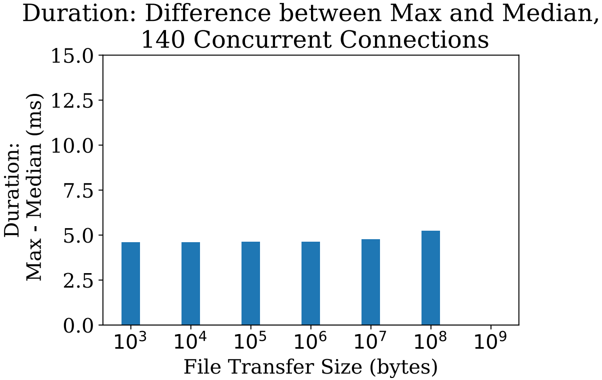


Time impact of recovery does not appear to depend on file size

EXPERIMENT II: RESULTS (CONTINUED)







Time impact of recovery does not appear to depend on file size *or number of concurrent connections*

DISCUSSION

- **What if the TCP connection which triggers a restore terminates before we have chance to remove it?**
  - Malicious process may still be removed if it references files not on the original filesystem
- **Possible to extend to other Operating Systems (besides Linux)?**
  - Blocker: TCP restore functionality
- **DoS potential?**
  - Use in conjunction with patching

LIMITATIONS

- **What if a restore is triggered but no policy matches are found?**
  - Fall back to start from original copy of FS
  - Connections are interrupted in this case
- **Doesn't verify validity of alerts**
  - Use public key cryptography to verify alerts using signing

FUTURE WORK

- **Dynamic Honeypot Creation**
  - Current work in dynamically creating two instances after infection: a honeypot and a restored version of the legitimate service
  - Dynamic "sanitization" of sensitive information on original container (see Araujo et al.)
- **Dynamic Assertions**
- **Verification of alerts**

CONCLUSIONS

- **Considers availability of service (including active connections)**
  - Able to maintain active connections even through recovery
- **Fast recovery and low overhead**
  - ~3 second for recovery in most cases
  - Only overhead is from LXC
- **Modular - can connect to virtually any IDS**
  - Recovery agent accepts JSON alerts from variety of sources
- **Available as open source**

THANK YOU

Questions?