Circumventing Forensic Live-acquisition Tools

- Rootkits for dubious defensive purposes
- Yonne de Bruijn (yonne.debruijn@os3.nl)
Digital Forensics

$ retrieve (evidentiary) data | form chain of evidence | prove stuff

  > like normal forensics, but digital...

$ non-volatile sources: persistent storage

$ volatile sources: network settings, RAM

$ offline (dead) acquisition: device is turned off

$ online (live) acquisition: device is turned on
Live-Acquisition

$ Reasons $

- Full Disk Encryption (FDE)
- Leave system running to reduce investigation “noise”

$ Process $

- order of volatility: first most volatile storage
- Windows: plethora of information
- Linux: …
Anti-Forensics (AF)

$ Increase difficulty of digital forensic process

$ or completely prevent

$ common techniques:

$ data hiding (slack space)

$ data destruction

$ encryption
“A rootkit is a tool that is designed to hide itself and other processes, data, and/or activity on a system” (Blunden, 2013)

Usually: persistent backdoors, root access, i.e. malicious stuff

Ring Zero Rootkit: highest privileges, can intercept commands from user-space

Hooking system calls: attach own code to system call
Research Question

$\$\text{What acquisition tools are available and used?}$

$\$\text{How can a system defend against those tools?}$

$\$\text{Can the tools or procedures be improved?}$
Forensics Wiki

$ imagers:

  ▶ dd
  ▶ or: dcflddl (forensic counterpart)
  ▶ or: dd_rescue
  ▶ or: pretty much any block-level copy tool
Silk Road

$ Online Drug Market

- Alleged owner arrested in 2013
- Sentenced to life, based on, amongst others: live-acquisition

$ Reddit users retrieved court transcripts

- suspect used Ubuntu + FDE → live-acquisition
- forensic toolkit: tar, dd, a camera running 40 minutes slow, and a good batch of ignorance
Dutch High Tech Crime Unit

$ dd

$ cp/tar, or other common copy tools

$ FTK/EnCase → standard

$ Encrypted evidence:
  > try publicly known exploits
  > other channels (maybe less secure in the past)
  > most effective? → “rubber hose” decryption

$ don’t often encounter AF
  > if they do → simple stuff
Problem

$ Tools run on suspect system:

› insecure environment

› use system tools —> might be patched to return garbage

› bring own tools —> might taint evidentiary system

› and... still using system kernel
Related Work

$ DDefy (Bilby, 2006) Windows rootkit

▷ defensive rootkit → not just for attacking?

▷ actively prevents \textit{dd} from acquiring certain files

$ Bunden (2009) warns for AF rootlets

$ Stüttgen & Cohen (2013) identified, exploited and patched issues in memory acquisition
Common Prevention

$ Check:

> /sysfs and /proc for loaded kernel modules (LKM)

> common signs of encryption \(\rightarrow\) implies don’t turn off device

> other scripts:

> sometimes used to null route logs, shred data

> check for known AF applications

> does not seem hardened against advanced rootkits
Putting it together

$ focus: tar, dd/dcfldd and FTK Imager
CLI → proprietary tool

$ goal: intercept tools and present different data, preferably without crashing them

$ weapon: ring zero rootkit

› easy to develop, could just as well run directly in kernel
Considerations

$ Control: must be hard to detect (no control application!)

› hook open system call and parse for magic control strings

$ Hide traces:

› hide fake data

› hide rootkit from /sysfs and /proc
Interception

$\text{Return fake data:}$

$\text{> } \text{tar: other user directory } \rightarrow \text{ framing}$

$\text{> } \text{imagers: from clean image located in filesystem}$

$\text{Trigger based, i.e. need a detection mechanism}$
Command Detection

$ Hook system calls used by tools

$ Parse calls for *magic strings*:

  › if(open(/dev/sda, params)

     › { fake = open(/clean, params);

     › return fake; }

$ Comparable for *tar*

$ Success?

Yonne de Bruijn
Demo
Linear Detection

$ Command detection not very flexible, easily broken

> better add some “pixie dust” to harden it!

$ What if we could detect behaviour…

> note: only implemented for dd/dcfldd
Demo
Success:

- detects and intercepts `dd/dcfldd`
- no data from `/dev/sda` present in output file!

Issues:

- output image is corrupted
- horrible effect on read/write speeds
- Some false-positives:
  - i.e. sometimes dumps fake MBR in `nano` editor
Prevention (1)

- Move acquisition to kernel:
  - no need for interception-sensitive system calls
  - direct access to virtual file system (VFS): vfs_read/vfs_write

- Encrypt communication between user-space and kernel-space (linux/crypt.h)
  - prevents parsing/changing the system calls
Prevention (2)

$ Problem: very rootkit minded

$ What if directly included in kernel $\rightarrow$ anti-forensic kernel

- need dedicated hardware solutions
- can not utilise kernel $\rightarrow$ directly talk to hardware

- so… how ‘bout hiding in firmware?
Conclusion (1)

$ Forensics toolkit: commonly available tools

$ Anti-forensic scenarios: many! in full control

$ Prevention

  > rootkits: create secure environment

  > tweaked kernel: difficult

  > hardware based acquisitions
Realistic threat to digital forensic process

- not yet seen in the wild
  - but technologically skilled attacker is certainly capable
Future Work

$ Extend linear detection to tar

$ Develop acquisition kernel-model

$ Implement code directly in kernel $\rightarrow$ anti-forensic kernel!

$ Debug, clean and expand
Questions?

source-code: https://bitbucket.org/yonne/atoolkit