JOHANNES KEPLER UNIVERSITY LINZ | JKU

Live-Forensik: Rootkit

Michael Sonntag

Institute for Information processing and microprocessor technology (FIM)

Johannes Kepler University Linz, Austria

sonntag@fim.uni-linz.ac.at

Scenario

- Investigating an image of a Linux system (CentOS 5.5) infected by a rootkit
- We will use live acquisition of data to gather information on this system
  - Simulated, i.e. we will produce a local file with the output
    - In reality this should be sent by netcat (nc) to a different computer to prevent modifications of the system under investigation!
- We will use various techniques to identify the problem
- The virtual machine is a VMWare image
  - Can also be opened in Virtualbox
- Under /mnt/cdrom there is the script “investigate.sh”
  - You can run it (“/mnt/cdrom/investigate.sh >report.txt”)
- But we are going to do the same (and some elements more) manually!

ATTENTION!

- This is part of a real rootkit, however it has been slightly modified
- It is **part** of a very old rootkit, so it is not that good/well hidden and very limited
  - Note the differences in output compared to the “real” binaries!
  - It has certain limitations which renders in relatively useless on modern systems
- BUT:
  - It is still a real rootkit
  - The source code is NOT available in the image (often it would have been compiled there, so it might still exist, perhaps only in parts of deleted files)
  - The binaries may NEVER be used anywhere else!

This is SOLELY for EDUCATIONAL USE!

Elements of good toolkits for live forensics

- Minimize system impact ✓ (apart from local log file)
  - Don't copy anything to the disk, binaries as small as possible
- Enforce the use of known binaries only
  Not possible: No statically compiled files (too complicated); use files on CD-ROM image for investigation
  - Make sure that no library from the system investigated is used
- Extensive logging and checksums ✓
  - Ensuring that no later modification can occur and that verification is possible
- No drivers needed for installing (→ CD-ROM better than USB!) ✓
  - Can be difficult → Depends on the system investigated
    - If very well secured, this might be difficult (IDS tries to prevent exactly this!)
- Copies data directly to another system (→ Network/Share or USB)
  Local file (too complicated)

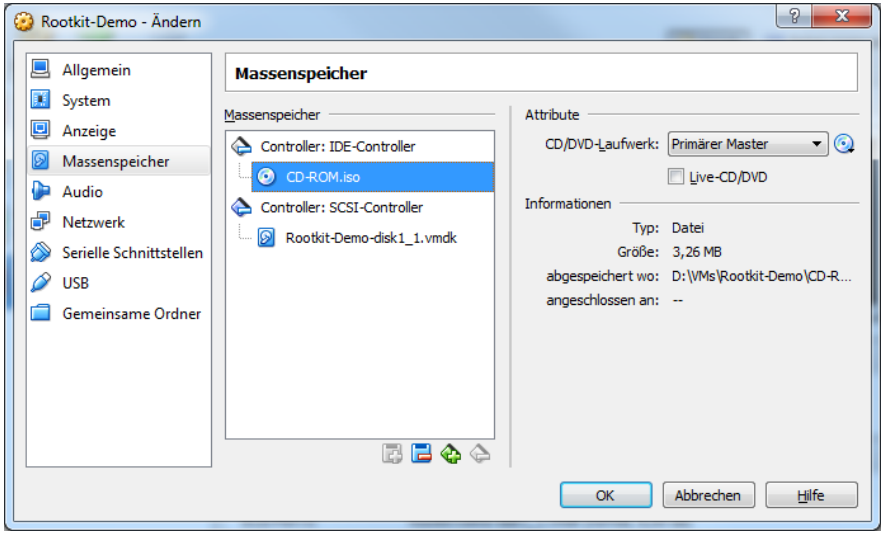
Basic information on the image

- CentOS 5.5 (=RedHat Enterprise Linux 5.5)
  - Basic/minimal installation: Commandline only (no GUI), no special applications installed
- Two users are available:
  - Username “user”, Password “user” (Normal user, no special permissions)
  - Username “root”, Password “root” (Administrator)
- Keyboard: German, but “\” is “shift-ü”
- Investigative tools: /mnt/cdrom/bin (needs to be mounted first!)
  - Perhaps useful: “export PATH=/mnt/cdrom/bin” to ensure to run only “our” programs
    - Note: Libraries will still be loaded from system; the tools are not statically compiled!

The image will be distributed in the class on DVDs/memory stick and can be downloaded from the website (Password for ZIP is disclosed in class)!

Importing the image

- Import the image from the DVD / Memory stick / download directory
- This image is the “original” system
- The forensic tools (not statically compiled; see previous slide) are located on an “external” CD-ROM: CD-ROM.iso
- Necessary process:
  - Import the virtual machine
  - Configure the virtual machine to use the CD image as the CD drive (see →)
  - Start the virtual machine
  - Mount the CD image:
    “mount /dev/cdrom /mnt/cdrom”



What we are not investigating here

- Copying RAM content
  - Difficult to do, investigation is very difficult and out of scope here!
- DNS cache
  - Not interesting here; problematic because of fixed file location
- No recovering deleted files still in use
  - The rootkit doesn't use such files

Basic information

- Generally: Try both commands - from “/mnt/cdrom/bin” as well as from the system - and compare both results!
- Date & time: “date”, “date –u”
  - Documenting the start of the investigation (incl. timezone)
- System: “hostname”, “uname –a”, “whoami”, “id”
  - Where are we? What kind of system is this? Who are we? (Last two not on "CD"!)
- Patch level: “rpm –qa”
  - Normally very late, as this is unlikely to change during the investigation!
- Uptime, logged in users: “w”, “who”
  - Are we alone (logins form network!)?
- Last logged in users: “last –a –i”
  - Including from where they logged in (here not interesting, but in general useful!)

IP/firewall information

- IP addresses: "ip addr"
  - Nothing special: localhost, IPv4 connection
  - sit0: Tunnel for IPV6 → IPv4
- IP devices: "ip link" and IP tunnels: "ip tunnel" show the same information
  - Take note of IP address/subnet → Might be necessary for "nc" (not used here)!
- Firewall configuration (iptables = standard on Linux)
  - "iptables-save":
    - Outgoing: No restrictions
    - Input and forwarding: A few default rules
      - Allowed: Local connections, ICMP (=pinging+…), ESP/AH: IPSec connections, UDP/224.0.0.251/5353 (Zeroconf/Multicast DNS), Port 631 (Internet Printing Protocol) Port 22 (SSH)

Network information

- ARP cache: “ip neigh show”
  - Useless here, as this system probably hasn’t connected anywhere
    - Note: Updates, software installation … might show some other systems
    - Depends also on the kind of network integration of the virtualization environment
- Routing table: “ip route show table all”
  - Current routes (here not very interesting)
- Routing cache: “ip route show cached”
  - Previous routes (here not very interesting)

Process information

- Processes: “ps aux”
  - Please take care: Which “ps” are your executing?
    - “/mnt/cdrom/bin/ps” or “/bin/ps” ?
  - Try both and compare them: What is strange?
    - Visual differences? Yes!
    - Content differences? Difficult because of the visual differences
      - We will come back to this later!
    - Count lines: “ps aux | wc –l” and “/mnt/cdrom/bin/ps aux | wc –l”
      - But: Perhaps the problem is not “ps” but “wc”? We don’t know yet!
        - Which wc did you use ☺?
        - Try “/mnt/cdrom/bin/ps aux | /mnt/cdrom/bin/wc –l”
- Anyway, we have found the first strange result!

Processes/ports (1)

- Listening: “netstat –an”
  - Three ports are open for listening:
    - UDP Port 68: BootP/DHCP (Waiting for info from DHCP server)
      - Does seem normal (depends on configuration!)
      - “cat /etc/sysconfig/network-scripts/ifcfg-eth0” → DHCP is really used/on
    - TCP Port 22: SSH server →Very common to be open on most systems!
      - Especially on commandline systems (otherwise: only console or telnet!)
      - Is a SSH server running? “ps aux | grep ssh”
        - Yes: /usr/sbin/sshd
        - Is this a “real” SSH server (or trojaned → Logging entered passwords)?

          Who knows, we would have to investigate more and in detail!

Processes/ports (2)

- Also open: Port 12345
  - This is a rather strange port: It is above 1024 and so should be a normal application
    - But no such application seems to be running?
  - What does Google say about port 12345?
    - Legitimate: NetBus remote administration tool for Windows
    - Often used for trojans, …
  - This looks very suspicious!
- But: We cannot get any more information out of this listing
  - So we keep it in memory and try to find out more!
    - Or use "netstat –anp"!

http://www.speedguide.net/port.php?port=12345

Open files/ports/…

- Showing all kinds of handles: “lsof –nP”
  - Attention: Very long output!
- So let’ focus a bit: “lsof –nP | grep 12345”
  - So this is the HTTP server running there! That looks a bit strange ….
  - Check whether such a server is really installed (init scripts/rpm are good starts!)
- What else is going on there: “lsof –nP | grep httpd”
  - It is running from the executable ”/usr/bin/httpd” and uses solely the C library
    - Plus the linux loader
  - It doesn’t have any other files open (try repeatedly) beside StdIn/StdOut/StdErr

Remote shell

- Try telneting there: “telnet localhost 12345” and “GET / HTTP/1.0<Ret><Ret>”
  - Doesn’t seem to be a webserver …
  - Try “ls –al;”
- This is a remote shell: If you can telnet there (firewalls!), you can issue any command, which will be executed as root
  - Note: Must be terminated by “;”, always returns an error message (low quality SW!)
  - Exiting the shell: “exit;”
- This is the first part of the rootkit!
- Try at home: Find out how it is started on boot
  - Hint: Check all kinds of init scripts!
- Note: Would this really be a problem here (=reachable from outside?)
  - Hint: See slides before ("iptables-save"!)

Back to ps

- Does “ps” show this program?
  - “ps aux | grep httpd” → No, but it should
- Does “/mnt/cdrom/bin/ps” show this program?
  - Yes it does!
- Result: “/bin/ps” doesn’t work quite all right, it probably was modified (“trojaned”)
  - We cannot trust its output any more

Checking file date/time

- We know the file “/bin/ps” has been modified - Can we find out the date/time?
- Date/times of a file: “stat /bin/ps” (Access omitted from output)
  - Modify: 31.3.2010 6:53
  - Change: 4.11.2011 13:59
- Compare this to the original date/times
    - How would we get at this? Install a “new” one in a virtual machine and check!
  - Modify: 31.3.2010 6:53
  - Change: 4.7.2010 5:42
- Result: The modification date seems to have been copied, but the change date is incorrect → The intrusion probably occurred on 4.11.2011 at about 14 o’clock
  - The rootkit installation program doesn’t work correctly regarding this!

Other information (OS, file system …)

- Kernel modules: “lsmod” and packages “rpm –qa”
  - Not very interesting here
- Mounted file systems: “mount –l”
  - Nothing mounted here apart from the system ones
- Free space: “df –k”
  - Not interesting here
- Scheduled jobs: “atq” → Nothing to show here
- System load: “top –bn 1”
  - Note: “httpd” is shown here → The rootkit doesn’t modify this commands’ output!

Checking the date of the intrusion

- Whet else happened on the system on 4.11.2011 13:59 (change time)?
- One possibility:

  “/mnt/cdrom/bin/find / -printf “%p;%Cx;%CT\n” | grep “11/04/2011;13:”

  — Why not the exact date? We don’t know whether this was the first or last action!

  — Too many results → we can still narrow it down!
- Also changed at about that time (apart from the directories they are in):

  — /bin/ps, /usr/bin/httpd: We already know them!

  — /bin/ls: That’s new! So some files seem to be hidden as well …

  — /usr/bin/chsh: That’s new!

  — Several other files (prelink-related, mails, yum cache, …)

    - Yum might potentially be interesting: Update check or who/what was installed?

/bin/ls: Hidden files

- Compare the output from our “find” commando to “ls” for the root directory
  - In practice this would be done by producing a full dump and automatic comparison
    - “Our” find and the one from the system itself
  - Here just use “ls -al /” and “find / -maxdepth 1”
- Result: “ls” has been “hacked” as it hides a directory otherwise existing!
  - Compare to previous slide – the suspicion there has been confirmed!
- There exists the directory “/rk” which shows up in “find” but not in “ls”
  - Check out its contents!

Rootkit files

- Now we find a different date: 3.11.2011 16:40
  - This could have been the time of the initial intrusion
- Check out the individual files
  - Find out what “fix” is for!
    - Try “strings fix” for a first view
  - Try to identify the content of the “backup” folder
  - What is “ptyp” and “ptyr” there for?
    - What can we learn from its content?
    - These are extremely important files: They show what is hidden!

What is “chsh”?

- Command to change the current/login shell
  - SetUID, modifies /etc/passwd → Very high permissions anyway
  - Main reason for trojans: If you get in as some user, you can become root through this
    - Drawback: You need the root password (NOT in trojaned versions!)
- This version has been modified, but how?
  - Try “strings chsh |more” → Can you see anything interesting?
  - No, the interesting parts (i.e. the “secret password”) has been hidden
    - Not that well, but good enough for this simple approach
  - Try the password (see next slide or the file README) – how is it to be used?
    - Enter it instead of a shell name and you receive a root shell
    - So to really test it, log in as “user” (check your rights with the command “id”)!

Rootkit password – Source code

- /* ROOTKIT_PASSWORD must be 6 letters due to my lame attempts at string hiding... */

  #define ROOTKIT_PASSWORD "rkdemo"

- char MAG[6];

  strcpy(MAG,"");

  MAG[0]=ROOTKIT_PASSWORD[0]; MAG[1]=ROOTKIT_PASSWORD[1];

  MAG[2]=ROOTKIT_PASSWORD[2]; MAG[3]=ROOTKIT_PASSWORD[3];

  MAG[4]=ROOTKIT_PASSWORD[4]; MAG[5]=ROOTKIT_PASSWORD[5];

  MAG[6]='\0';

- Password is stored in executable as separate characters
  - If you know this, you can see it clearly in the output of “strings chsh” as well!
- Practice: Deassembly/Decompilation or Debugging

Conclusions (1)

- Generally the investigation would be more difficult,
  - especially for files:
    - Using external tools and producing a log
    - Using the internal tools and producing a log
    - Comparing those files in a spreadsheet/diff/…
    - Find out date/time of files at the time of intrusion (if known …)
  - and binaries:
    - Install a “new” version in a virtual machine
    - Bring it to exactly the same patch level/software
    - Compare md5 values of files in both VMs to find out which ones were modified
      - Reinstall might be easier!

Conclusions (2)

- What we didn't find out:
  - How the intrusion took place
    - When it took place → We have only some hints, but which are quite good
  - What the attacker was after (but: no interesting content here anyway ☺)
  - Complete list of changes: Have we really found everything?
    - Probably yes, but some parts might also have been hidden better!

Thank you for your attention!

Michael Sonntag

Institute for Information processing and microprocessor technology (FIM)

Johannes Kepler University Linz, Austria

sonntag@fim.uni-linz.ac.at