R_: The Exponential Growth of Rootkit Techniques

by

Jamie Butler

Bill Arbaugh

Nick Petroni

KOMOKU

Agenda

Definitions

Emerging trends

Historical rootkits

Corporate rootkits

Rootkit techniques

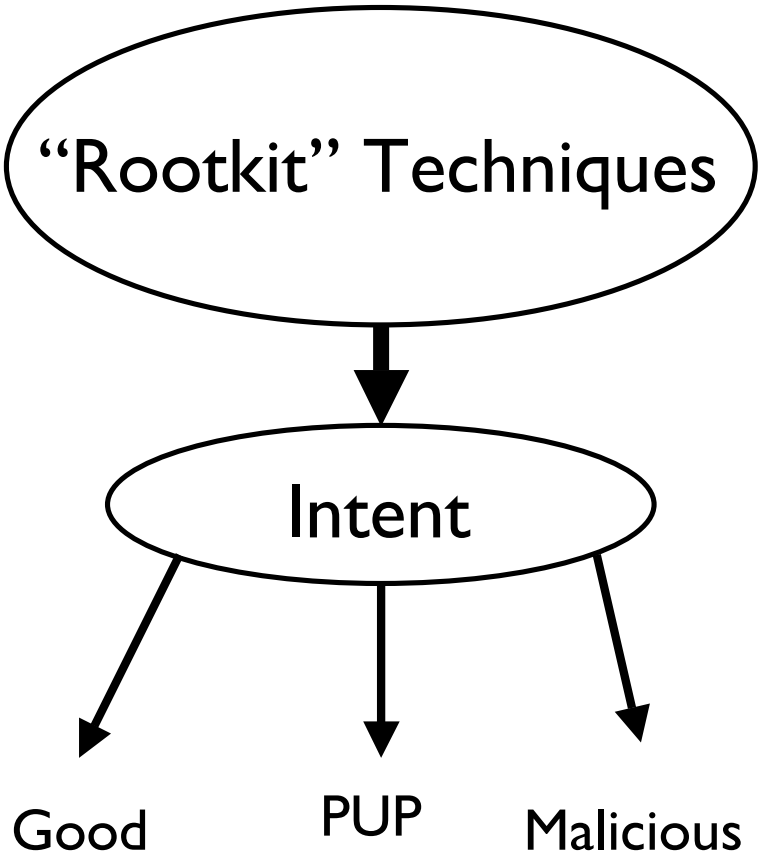
Rootkit detection

Conclusion

Definition of a Rootkit

Software that hides itself or other objects, such as files, processes, and Registry keys, from view of standard diagnostic, administrative, and security software. - Mark Russinovich

Malicious and good programs can use the same techniques- the difference is intent!



KOMOKU

Categorization as malicious depends on intent and approval

Good vs. Evil intent

Intent is predicated upon

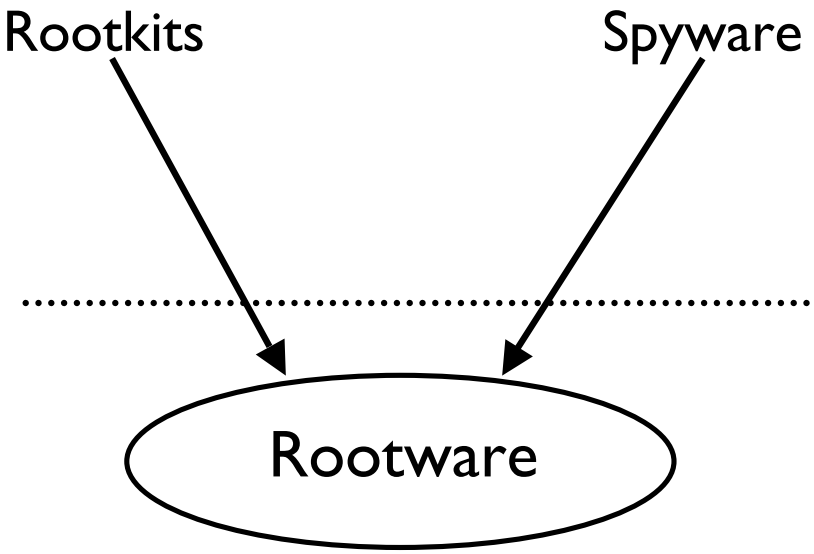
- Informed consent of the owner or user of the computer system
- Approval of the owner or user of the computer system

Rootkits vs. Spyware

Rootkits hide resources

Spyware collects data

Rootware: software that collects data without the knowledge and consent of the owner of the system and that employs stealth to further that ignorance



Rootware in the Wild

MiniKeylogger*

Monitors keystrokes, file operations, dial-up activities, and Internet Explorer

Hides process and service using a driver

Powered Keylogger*

Logs keystrokes, mouse clicks, passwords, web-activities, e-mail activities, screen shots, and idle status.

Hides all files, directories, Registry entries, and processes it creates using a driver

* Source: Symantec Security Response
http://securityresponse.symantec.com

KOMOKU

Historical Background

- Cuckoo's Egg
  - Late 1980's
  - Access discovered because of a 75 cent accounting imbalance
- Hiding Out Under UNIX by Black Tie Affair
  - Circa 1989
  - Phrack Volume Three, Issue 25
  - Hid tty from "who"

Historical Background

SunOS 4.x Rootkit

- CERT Advisory CA-1994-01
- Replaced system files – ps, ls, du, login, netstat

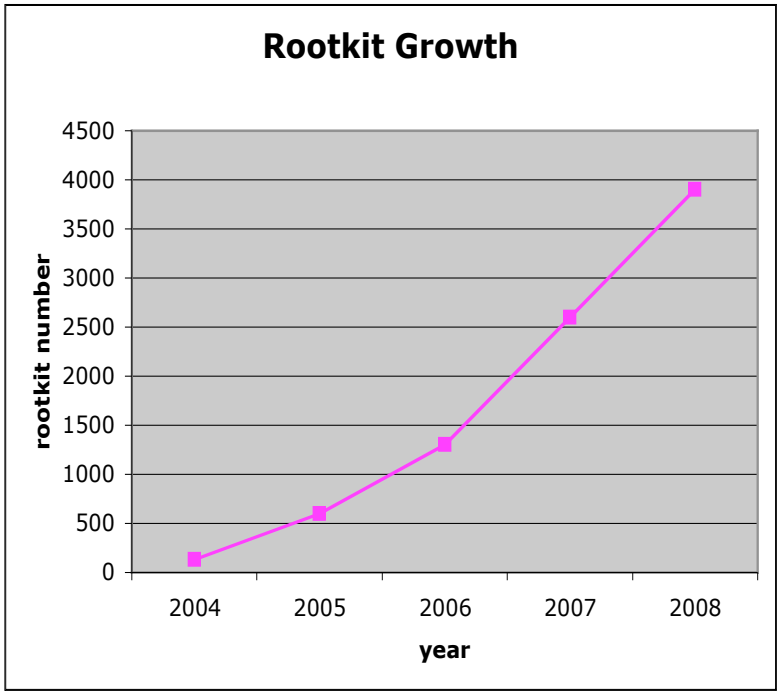
Windows NT Rootkit

- Circa 1999 – Greg Hoglund
- Hooked the kernel in memory

Recent Explosion

400% growth in the number of rootkits recorded by McAfee from 2004 to 2005

McAfee predicts an annual growth rate of at least 650% over the next two or three years for the current Windows architecture



Rootkit Techniques

One way to hide is to gain control of execution
- Replace system programs
- Hooks
- Callbacks
- Specialized registers
- Layered drivers
- Others

Another way to hide is to manipulate kernel data itself
- Lists of processes, drivers, etc.
- Handle tables
- Others

Virtualization

Hooks

Inline function hooks – similar to hotpatching

- Part of original function is overwritten with an instruction that causes a change in execution

System call hooks – used in the NT Rootkit

- Replace addresses of functions within a table provided by the kernel

Import Address Table (IAT) hooks – used in many user land rootkits

- Replace address of imported function from one DLL with the address of a different function

KOMOKU

Hooks

IDT hooks – used in the Shadow Walker Rootkit

- Replace in the table that handles interrupts, the IDT, the Interrupt Service Routine (ISR) address with the address of a rootkit function
- IRP table hooks
  - All drivers have a function table corresponding to the different form of I/O Request Packets (IRPs) the driver will handle
  - Replace the addresses in the original IRP table

KOMOKU

Callbacks

A callback is a documented way to gain execution control by registering a function to be “called back” when a certain event occurs on the system. Callback registration is exposed by the operating system in the form of an API.

Examples

- Windows Message Hooks -most keylogging spyware
- File system callbacks
- Object Manager functions
- NDIS protocols

KOMOKU

Special Registers

Debug registers

- Can be used to gain execution when execution reaches a certain point or when there is a memory access on a particular location

Model Specific Registers (MSRs)

- Can be used to gain control when certain instructions are executed such as SYSENTER
- Similar to an IDT hook

KOMOKU

Layered Device Drivers

Allows the interception of all I/O

- Keyboard
- File system
- Network

Direct Kernel Object (data) Manipulation

Unlink the list of processes and drivers

- FU Rootkit

Alter handle tables

- FUTo Rootkit

KOMOKU

VM Rootkits

SubVirt

- University of Michigan and Microsoft Research
- Works against VMWare and Virtual PC
- Places the OS within a virtual machine
- Modifies the boot sector for persistence

Blue Pill

- Joanna Rutkowska
- Relies upon AMD SVM Technology

Current Rootkit Detection Methods

Heuristic or Behavioral Detection

Integrity Detection

Signature Based Detection

Diff Based or Cross View Detection

Examples

Integrity Checkers – Komoku and SVV

Signature Scanners – AV Products

Behavior and Cross View Based Approaches

- Microsoft Strider GhostBuster
- SysInternals' Rootkit Revealer
- RAIDE

Integrity Types

Bitwise Integrity Checks

- Komoku and SVV

Semantic Integrity Checks

- Komoku

Bitwise Integrity

Detects unauthorized changes to system files or to loaded OS components in memory.

Uses a baseline database containing their hash values

Periodically calculates and compares the hashes of these files against the trusted baseline.

- Example: Komoku and SVV
  - Verifies that immutable code and data in system memory has not been altered
  - Very powerful but difficult to do right

KOMOKU

Semantic Integrity

Unlike bitwise integrity which only ensures that scalar bit values remain invariant, Semantic integrity ensures that a first order logic predicate remains invariant.

This enables detection of DKOM and other sophisticated attacks that target the structures within an operating system.

Signature Based Detection

"Fingerprint Identification"

Searches memory or the file system for unique byte patterns (signatures) found in the rootkit's code.

Tried N' True Approach - Has been used by AV scanners for many years.

Highly accurate, but ineffective against unknown rootkit / malware variants (for which a signature does not exist) or deliberately obsfucated code.

KOMOKU

Behavioral Detection

Attempts to detect the effects of a rootkit on the victim system which means it may detect previously unknown rootkits.

- Detecting diverted execution paths.
  - Deviations in executed instructions – PatchFinder by Joanna Rutkowska
- Detecting alterations in the number, order, and frequency of system calls.

May suffer from a high false positive rate.

- Most end users don’t have the skill to screen out false positives.

Cross View Based Detection

Uses two views of same information

- Example:
  - Walk the list of EPROCESS structures in memory
  - Call ZwQuerySystemInformation
  - Compares results
  - Any differences are reported

Often uses undocumented structures

Occasionally uses “clean” and “dirty” boots for comparison

What makes detection so hard?

- Determining intent – good vs. evil
  - Many third party security add-ons employ the same methods as rootkits
  - Example
    - UAY Rootkit
    - Zone Alarm

Conclusion

Questions?