

# Using Rootkit Technology for Honeypot-Based Malware Detection

Steve Hanna

ICSI Summer 2007 Networking Intern  
CCEID Meeting, September 2007

# GQ and Limitations

- GQ is a honey farm that currently can automate honeypot infection and detection of worm-like traffic.
- A network-based detection mechanism is in place.
- The class of detectible malware is small and must exhibit certain patterns.
- VMWare based solution requires constantly resetting machine state, regardless of the success of the attack.
- How can we improve this?

# Goals

- We would like to detect a larger class of malware, not restricted to just worms.
- Our solution must scale.
- The chosen mechanism must allow accurate reporting of the malicious activity.
- The system must respond gracefully to recover quickly from failed attacks.
- If the target computer was not infected by the malicious activity, continue as normal without resetting machine state.

# Goals and the Approach

- Design a rootkit solution to monitor system activity.
- Send supposedly malign traffic to a remote location for logging and processing.
- If analysis determines that a side effect took place on the machine as a result of some external action, log this data and reset the machine state.

# Design Specifics

- Internal VM Sensor is comprised of three components.
  - Kernel Driver
  - Layered Service Provider (LSP)
  - White List and Forwarding Mechanism (WLFM)
- The LSP and WLFM live in user space, the driver lives in kernel space.

# Design Specifics: Kernel Driver

- Hooks System Service Descriptor Table (SSDT) to monitor file system, registry and process creation.
- When one of these classes of actions takes place, the details of the function called are logged to an internal kernel list.
- The driver exposes a file system object to communicate with the user land components.
  - When the WLFM requests data, if the list is not empty the entry is transferred to user land.

# Design Specifics: WLFM

- Component has multiple functions
  - Collect operating system activity through the file system object while not exposed to a network. This is referred to as the white list.
  - Read operating system activity while running, if not on white list, forward to a specific a logging mechanism.
- Logging mechanism is currently very noisy, generates a lot of data and simple filtering is not the solution.

# Experimenting with the Monitors

- Large collection of malware obtained from various sources.
- A local (closed) network was setup on one machine using multiple VMWare instances.
- Consisted of three machines.
  - Logging - Patched Windows XP SP2
  - Monitor - Unpatched Windows XP Original
  - Initial Infection - Unpatched Windows XP Original

# Experimenting Continued

- Much to my surprise, I could not get any of the malware to work!
- So, I wrote my own worm to show the capabilities of the system.
- Ideally, show writing malware to disk, observing execution, watching socket activity.

# Worm Details

- Consisted of a simple client program that listened on a UDP port.
- Whatever shellcode was sent to this port was executed. Malware sent single UDP packet to random addresses on local network.
- Using tftpd32.exe as the propagation mechanism, the executed code uses tftp.exe to grab a copy of the daemon and a copy of the malware.
- Executed the malware, ad nauseam.
- We observed all of the above actions within the logged data.

# Limitations: Broad

- There is a lot of background noise.
- A better filter/whitelist would mean that if any traffic was recorded, we would need to take note.
- Currently, no heuristic to actually make the determination if an infection was successful.

# Limitations: Specific

- Works on all versions of Windows XP, however it remains untested on Vista.
- Userland program polls file system object, not optimal.
- Mechanisms to specifically detect and prevent attacks against our code are not implemented. However, this is acknowledged as an arms race issue.

Questions?