UCF Stands For Opportunity

CAP6135: Malware and Software Vulnerability Analysis

The Next Generation Peer-to-Peer Botnet Attacks

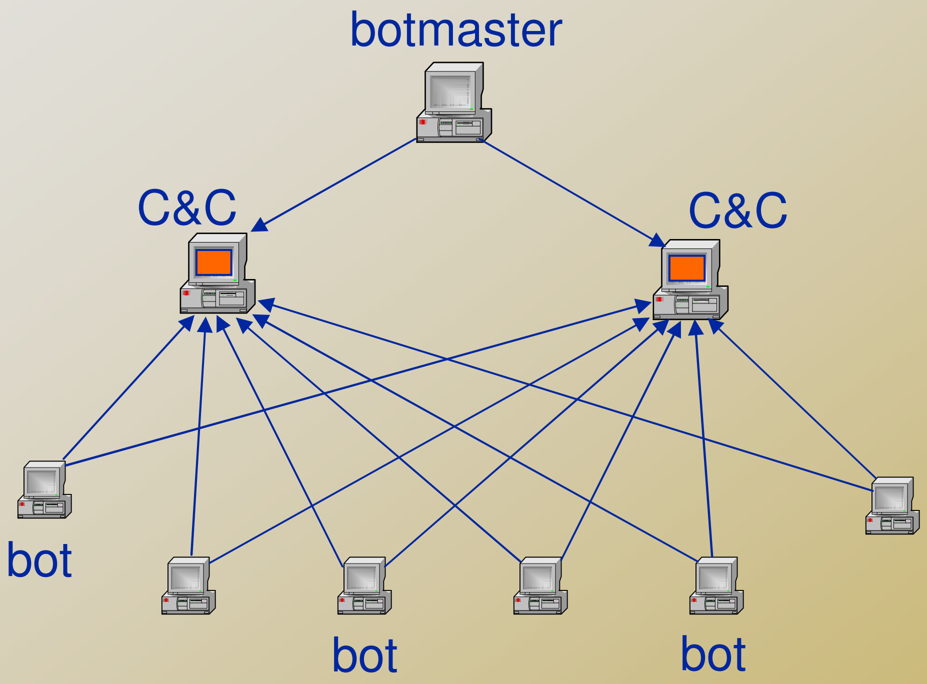
Cliff Zou

Spring 2009

What Is a Botnet?

- Botnet: bot + network
  - Bot: compromised machine installed with remote controlled code
  - Networked bots under a single commander (botmaster, botherder)
- Botnet is the major threat nowadays
  - Large-scale worm attacks are old news
  - Profit: motivation for most attackers
    - Spam, phishing, ID theft, DoS blackmail
    - Botmaster with thousands of machines at command has attack power

Current Botnet Command & Control Architecture



- Bot periodically connects to one/some of C&C servers to obtain command
  - Hard-coded IPs or DNS names of C2 servers
- C&C: usually Internet Relay Chat (IRC) based

Motivation

- Most works target current botnets only
  - Rely on current botnet′s architecture, infection methods, and control network
    - Study current botnets is important, but not enough
  - May not work if botmasters upgrade their future botnets
    - E.g., recent Peacomm and Storm botnet --- basic P2P botnets
  - We must study one step ahead
    - How botnets will evolve?
    - How to defend future botnets?

Three Possible Moves of Future Botnets

- Peer-to-peer structured botnets
  - More robust C2 architecture
  - We present a hybrid P2P botnet
- Honeypot-aware botnets
  - Honeypot is popular in malware defense
  - A general principle to remove inside honeypot spies
- Stealthy botnets
  - Keep bots as long as possible
  - We study “rootkit” techniques

Peer-to-Peer Botnet

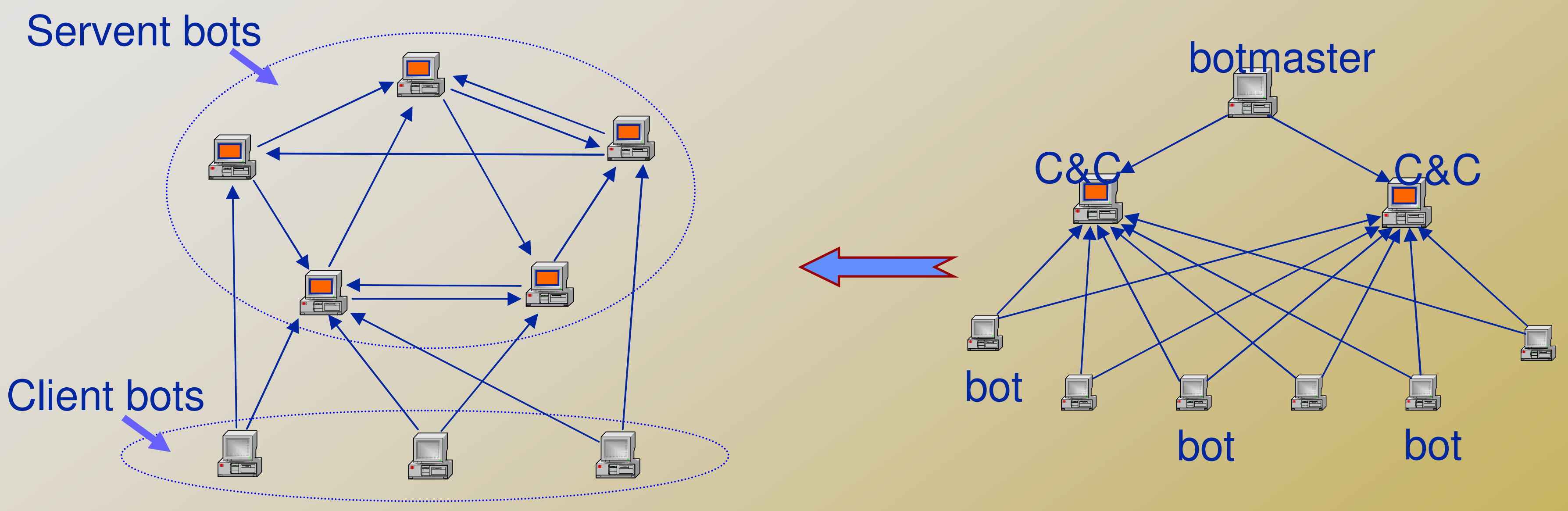
Peer-to-Peer (P2P) based Control Architecture?

- Weakness of C&C botnets
  - A captured bot (e.g., honeypot) could reveal all C2 servers
  - The few C2 servers can be shut down at the same time
  - A captured/hijacked C2 server could reveal all members of the botnet
- C&C centralized → P2P control is a natural evolution
  - P2P-based network is believed to be much harder to shut down

P2P upgrade is not so simple for botnets

- Current P2P protocols are not designed for the purpose of botnets
  - Easy exposure of botnet members
    - E.g., query to obtain response, P2P crawlers
  - Excess traffic susceptible to detection
  - Bootstrap process against the design goal
    - The few predefined bootstrap nodes have the same weakness as C&C servers
- Botmasters need easy control/monitor of their botnets
  - Understand botnet size, distr., bandwidth, etc.

Proposed Hybrid P2P Botnet



- Servent bots: static IPs, able to receive incoming connections
  - Static IP ensures a stable, long lifetime control topology
- Each bot connects to its "peer list"
  - Only servent bot IPs are in peer lists

Dramatically increase the number of C&C servers

Botnet Command and Control

- Individualized encryption key
  - Servent bot *i* generates its own symmetric key K_i
  - Any bot connecting with bot *i* uses K_i
    - A bot must have (IP_i, K_i) in its peer list to conect bot *i*
- Individualized service port
  - Servent bot *i* chooses its port P_i to accept connections
  - A bot must have (IP_i, K_i, P_i) in its peer list to connect bot *i*
- Benefits to botmasters:
  - No global exposure if some bots are captured
  - Dispersed network traffic
  - Go through some firewalls (e.g., HTTP, SMTP, SSH holes)

Botnet Monitor by Botmaster

- Botmasters need to know their weapons
  - Botnet size
  - bot IPs, types (e.g., DHCP ones used for spam)
  - Distribution, bandwidth, diurnal …
- Monitor via ***dynamical*** sensor
  - Sensor IP given in a monitor command
  - *One sensor, one shot, then destroy it*
    - Use a sensor's current service to blend incoming bot traffic

P2P Botnet Construction

- Botnet networked by peer list
- Basic procedures
  - New infection: pass on peer list
  - Reinfection: mix two peer lists
    - Ensure balanced connectivity
- Remove the normal P2P bootstrap
  - Or, increase entries in bootstrap as botnet grows

P2P Botnet Construction

- OK? No!
  - Real botnet is small compared to vulnerable population
    - Most current botnet size ≤ 20,000
    - Reinfection happens ***rarely***
  - Not balanced topology via new infection only
- Simulation results:
  - 500,000 vulnerable population
    - Botnet stops infection after reach 20,000
  - Peer list = 20, 21 initial servent bots, 5000 bots are servent bots
  - Results:
    - < 1000 reinfection events
    - Initial servent bots: > 14,000 in-degree
    - 80% of servent bots: < 30 in-degree

P2P Botnet Construction

- Peer-list updating procedure
  - Obtain current servent bots information
  - Request every bot connect to a sensor to obtain a new peer list
- Result: all bots have balanced connectivity to servent bots used in this procedure
  - Use once is enough for a robust botnet
  - Can be used to reconnect a broken botnet

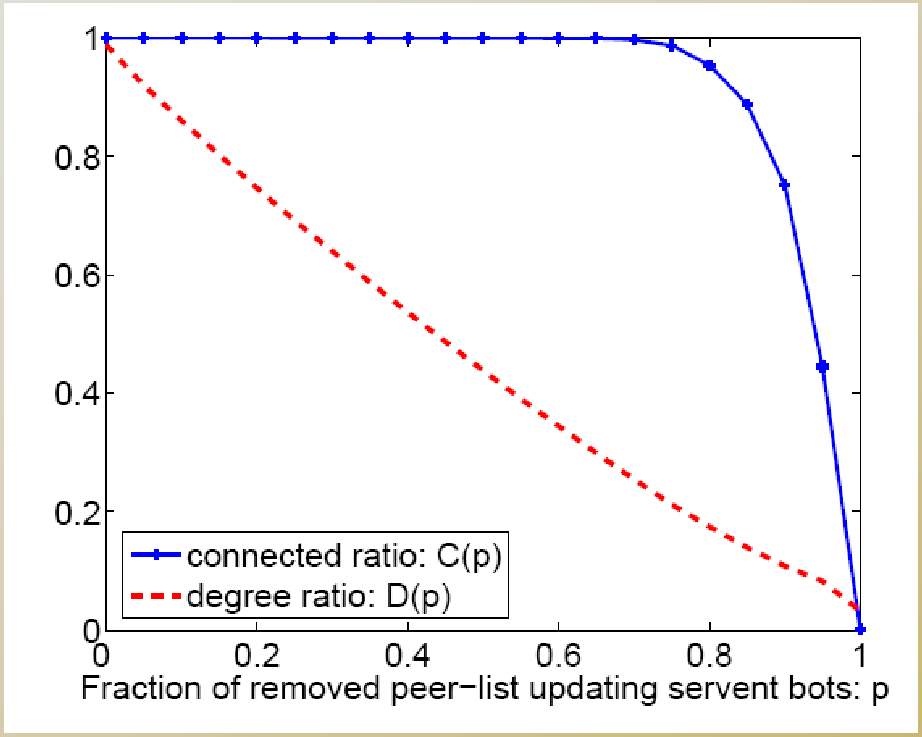
Robustness Metrics

- What if top *p* fraction of servent bots are removed?
  - Removed due to: defense, diurnal, link failure…

$$C(p) = \frac{\#\text{ of bots in the largest connected graph}}{\#\text{ of remaining bots}}$$

$$D(p) = \frac{\text{Avg. degree of the largest connected graph}}{\text{Avg. degree of original botnet}}$$

Botnet Robustness Study



- 500,000 vulnerable population, botnet = 20,000
- Peer list = 20, 5000 bots are servent bots
- Run peer-list updating once when having 1000 servent bots

Defense Against the Botnet

- Shut down a botnet before the first peer-list updating procedure
  - Initial servent bots are the weak points at beginning
- Honeypot based defense
  - Clone a large set of “servent” bots
    - But it can survive with only 20% servent bots left
  - Obtain peer lists in incoming infections
- Forensic analysis of botmaster’s sensor
  - Challenge: Log of unknown port service and IP beforehand

What about Existing P2P Protocols?

- Existed P2P botnets: Peacomm, Storm
- Built on Overnet protocol
  - Distributed Hash Table (DHT)-based

- Has a predefined list for initial bootstrap
  - Could be centralized point of failure
    - Defend by shutting down the list at the early stage

Index Poisoning Attack

- A bot queries one of 32 predefined indexes to find command
  - Botmaster publishes command via these indexes
  - Problem: “index poisoning attack”
    - Defenders publish many more of these indexes
    - Real command indexes are hard to find
    - Discussed in a LEET’08 paper
  - It is a fundamental problem for publish/subscribing P2P networks

A Simple Solution to Index Poisoning Attack (ongoing work)

- Observation of P2P botnets:
  - Only command index needs to be published; why allow arbitrary bot to publish?
- Index authentication
  - Bot is hard-coded with public key K^+
    - K^- is known only to the botmaster
  - A command m is published as $K^-(m)$
  - Any bot drops an index announce or query response if it does not contain $K^-(m)$
- Only a small module addition to existing P2P protocol/program

Honeypot-Aware Botnet

Honeypot-Aware Botnet

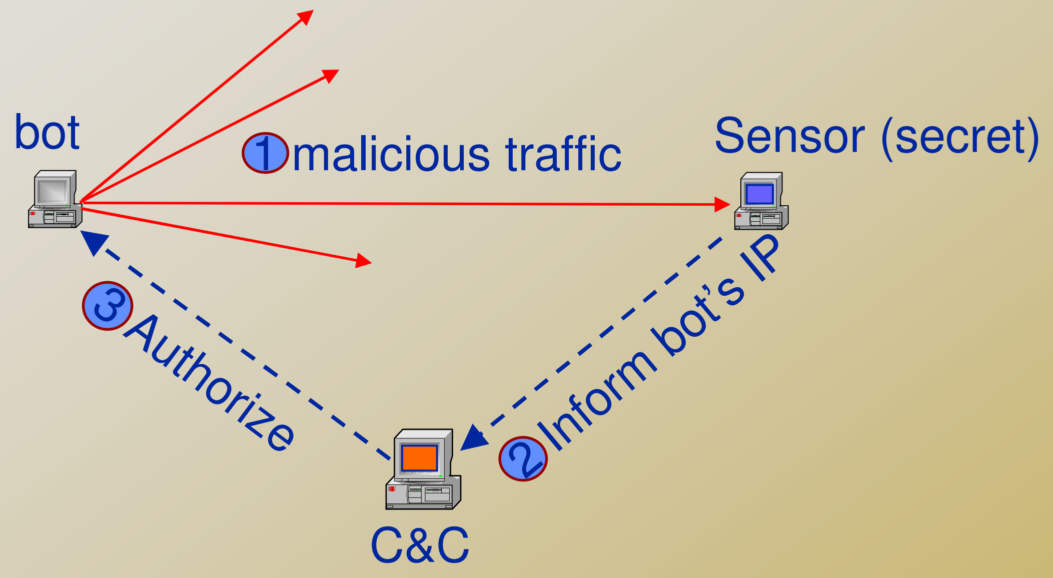
- Honeypot is widely used by defenders
  - Ability to detect unknown attacks
  - Ability to monitor attacker actions (e.g., botnet C&C)
- Botnet attackers will adapt to honeypot defense
  - When they feel the real threat from honeypot
  - We need to think one step ahead

Honeypot Detection Principles

- Hardware/software specific honeypot detection
  - Detect virtual environment via specific code
    - E.g., time response, memory address
  - Detect faculty honeypot program
  - Case by case detection

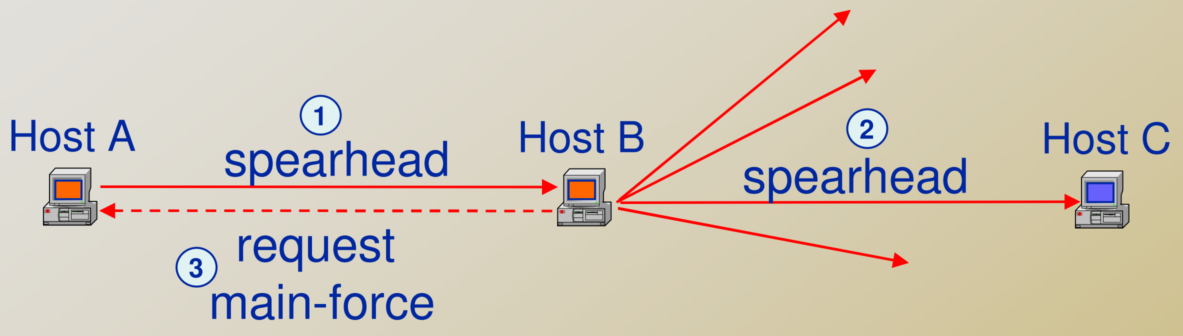
- Detection based on fundamental difference
  - Honeypot defenders are liable for attacks sending out
    - Liability law will become mature
    - It's a moral issue as well
  - Real attackers bear no liability
    - Check whether a bot can send out malicious traffic or not

Detection of Honeypot Bot



- Infection traffic
  - Real liability to defenders
  - No exposure issue: a bot needs to do this regardless
- Other honeypot detection traffic
  - Port scanning, email spam, web request (DoS?)

Two-stage Reconnaissance to Detect Honeypot in Constructing P2P Botnets



- Fully distributed
  - No central sensor is used
  - Could be fooled by double-honeypot
    - Counterattack is presented in our paper
- Lightweighted spearhead code
  - Infect + honeypot detection
  - Speedup UDP-based infection

Defense against Honeypot-Aware Attacks

- Permit dedicated honeypot detection systems to send out malicious traffic
  - Need law and strict policy
- Redirect outgoing traffic to a second honeypot
  - Not effective for sensor-based honeypot detection
- Figure out what outgoing traffic is for honeypot detection, and then allow it
  - It could be very hard
- Neverthless, honeypot is still a valuable monitoring and detection/defense tool

Stealthy Botnet using Rootkit Techniques

Motivation

- Botmaster wants to keep bots as long as possible
  - Require bot code to avoid detection
- Rootkit: Malicious code hiding techniques
  - E.g., change running process display
  - Make changes to the host OS
    - Hooking (Hacker Defender & NT Rootkit)
    - Direct Kernel Object Manipulation (FU)
    - Memory Subversion (Shadow Walker)
  - Changes in OS can be detected

OS Independent Rootkits

- Subvert system without making changes to the host OS
  - Hardware Virtualization Rootkits
    - Bluepill (AMD) – Joanna Rutkowska
    - Vitriol (Intel) – Dino A. Dai Zovi
  - BIOS Rootkits
    - Proof of concept ACPI BIOS Rootkit – John Heasman
  - **Chipset level Network Backdoor** [AsiaCCS'09]
    - Interacts directly with network card
  - **SMM Rootkits** [Securecomm'08]
    - SMM: System Management Model (Intel processors)
  - Both are possible for high-valued botnets

Chipset Level Network Backdoor

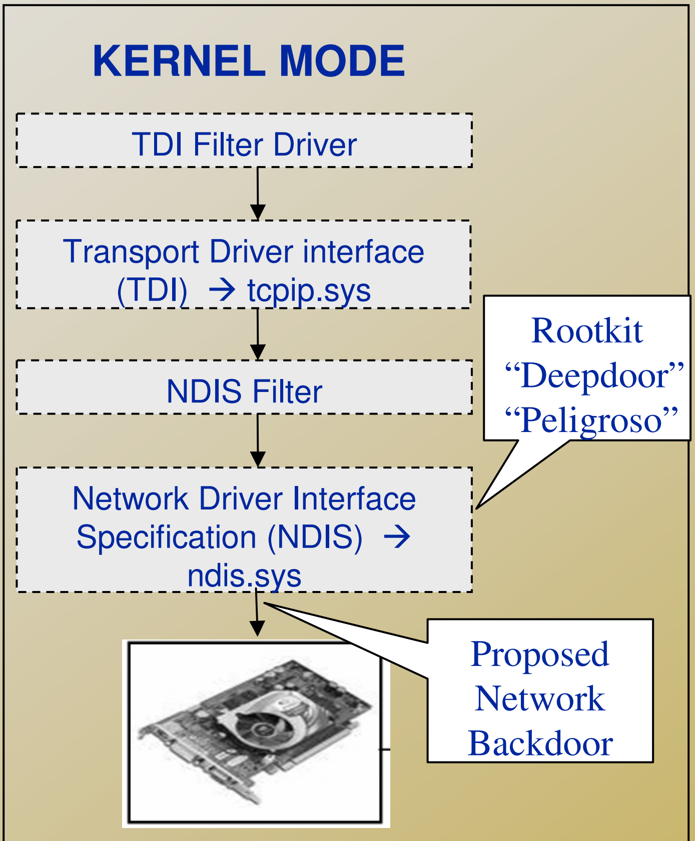

KERNEL MODE
TDI Filter Driver
Transport Driver interface (TDI) → tcpip.sys
NDIS Filter
Network Driver Interface Specification (NDIS) → ndis.sys
Rootkit "Deepdoor" "Peligroso"
Proposed Network Backdoor

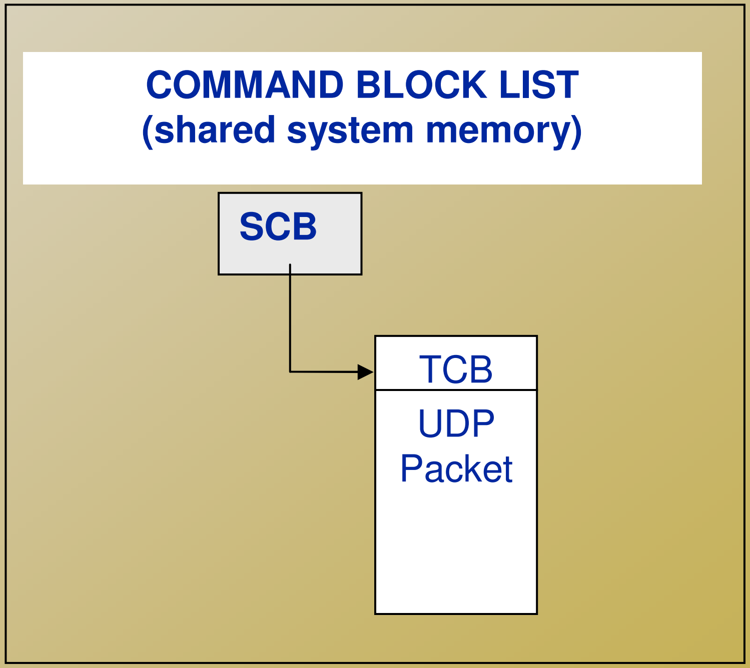
Figure 1: Windows Network Architecture

Network Backdoor

- Surprisingly easy… We just need to write to a few registers on the network card (also located in the PCI configuration space)
- Developed for Intel 8255X Chipset
  - Tested on Intel Pro 100B and Intel Pro 100S cards
  - Lots of other cards compatible with the 8255X chipset
  - Open documentation for Intel 8255X chipset

Data Exfiltration – Sending data out

1. Build A Transmit Command Block (TCB)
2. Build the data packet
3. Check that the LAN Controller is idle
4. Load the physical address of the Transmit Command Block into the System Control Block
5. Write CU_start into the System Control Block to initiate packet transmission



Why is SMM attractive to rootkits?

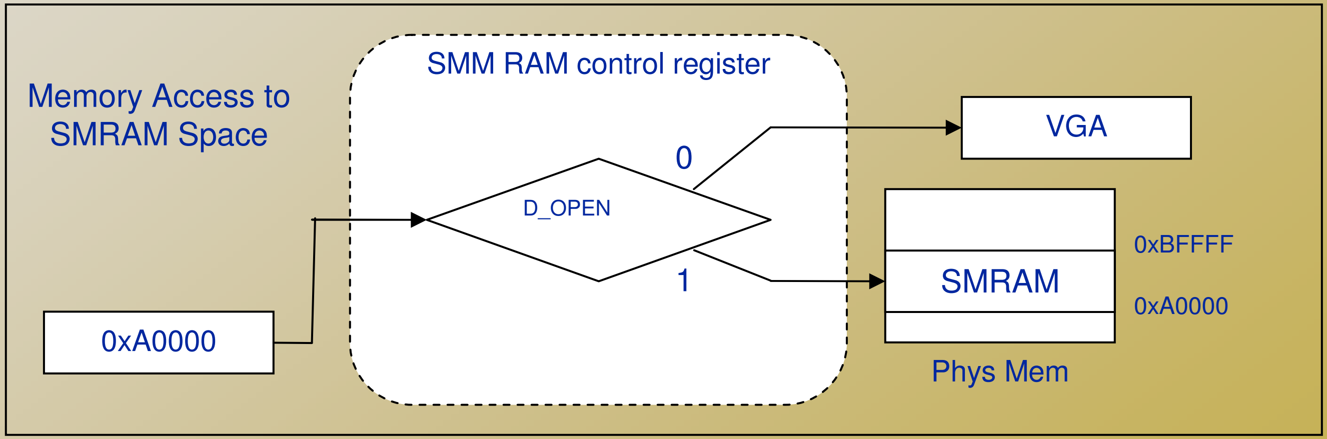
- SMM: originally for managing low-level hardware operations
- Isolated memory space and execution environment that can be made invisible to code executing in other processor modes (i.e. Windows Protected Mode)
- No concept of “protection”
  - Can access all of physical memory
  - Can execute all instructions, including privileged instructions
- Chipset level control over peripheral hardware
  - Intercept interrupts without changing processor data structures like the IDT
  - Communicate directly with hardware on the PCI bus

SMRAM Isolation

- SMRAM isolation is enforced by D_OPEN bit in SMM RAM control register (SRAMC)
  - D_OPEN=0, access VGA; D_OPEN=1, access SMRAM

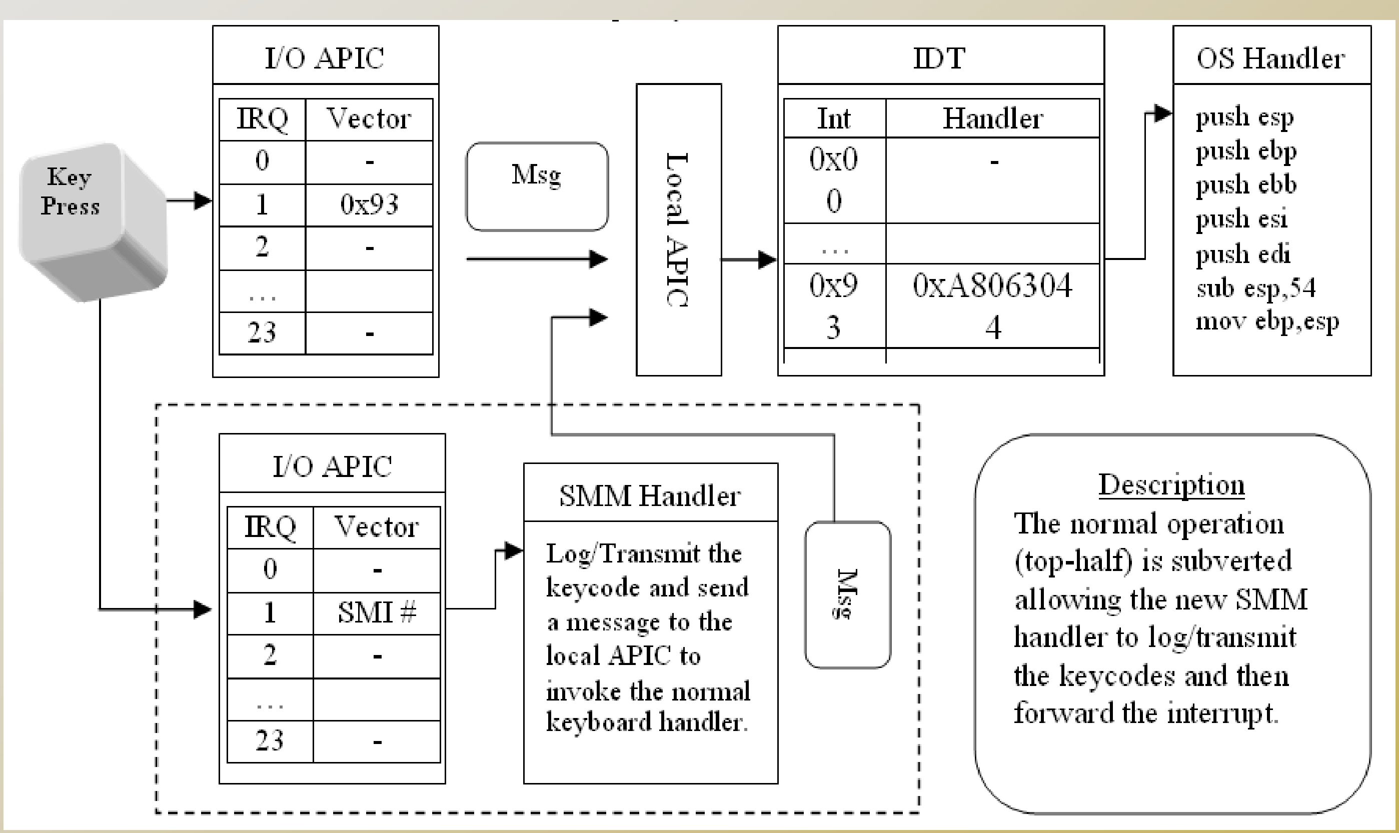
Res.	D_OPEN	D_CLS	D_LCK	GLOBAL SMRAME	0	1	0

- If D_LCK bit in SRAMC is set, this register becomes read only
  - After installing, SMM rootkit set D_LCK to prevent others to access SMRAM



- Rootkit Installation Procedure
  - Make SMM visible (D_OPEN=1)
  - Opening SMRAM for Writing
  - Writing in a new SMM handler
  - Make SMM invisible (D_OPEN=0)
  - Lock SMM (D_LCK=1)
- Only documented way to clear D_LCK is via a reset

Chipset Level Keylogger



I/O APIC

IRQ	Vector
0	-
1	0x93
2	-
…	
23	-

IDT

Int	Handler
0x00	-
…	
0x93	0xA8063044

OS Handler

```
push esp
push ebp
push ebb
push esi
push edi
sub esp,54
mov ebp,esp
```

I/O APIC

IRQ	Vector
0	-
1	SMI #
2	-
…	
23	-

SMM Handler

Log/Transmit the keycode and send a message to the local APIC to invoke the normal keyboard handler.

Description

The normal operation (top-half) is subverted allowing the new SMM handler to log/transmit the keycodes and then forward the interrupt.

Sending out Key Logs

- Using network backdoor
- Rootkit in SMM directly interact with network card to send out data
  - Network backdoor can also receive data for possible botmaster's command
  - Details see our paper

Summary

- We have to be well prepared for future botnets
  - Only studying current botnets is not enough

- It is an ongoing war between botnet attacks and defenses

References on P2P Botnet Research

- Ping Wang, Sherri Sparks, and Cliff C. Zou, An Advanced Hybrid Peer-to-Peer Botnet, *HotBots*, 2007.
- R. Vogt, J. Aycock, and M. Jacobson, Jr. Army of Botnets, *NDSS*, 2007.
- G. Starnberger, C. Kruegel, and E. Kirda. Overbot - a botnet protocol based on kademlia. *SecureComm*, 2008.
- J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study, *HotBots*, 2007.
- T. Holz, M. Steiner, F. Dahl, E. W. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. *LEET*, 2008.