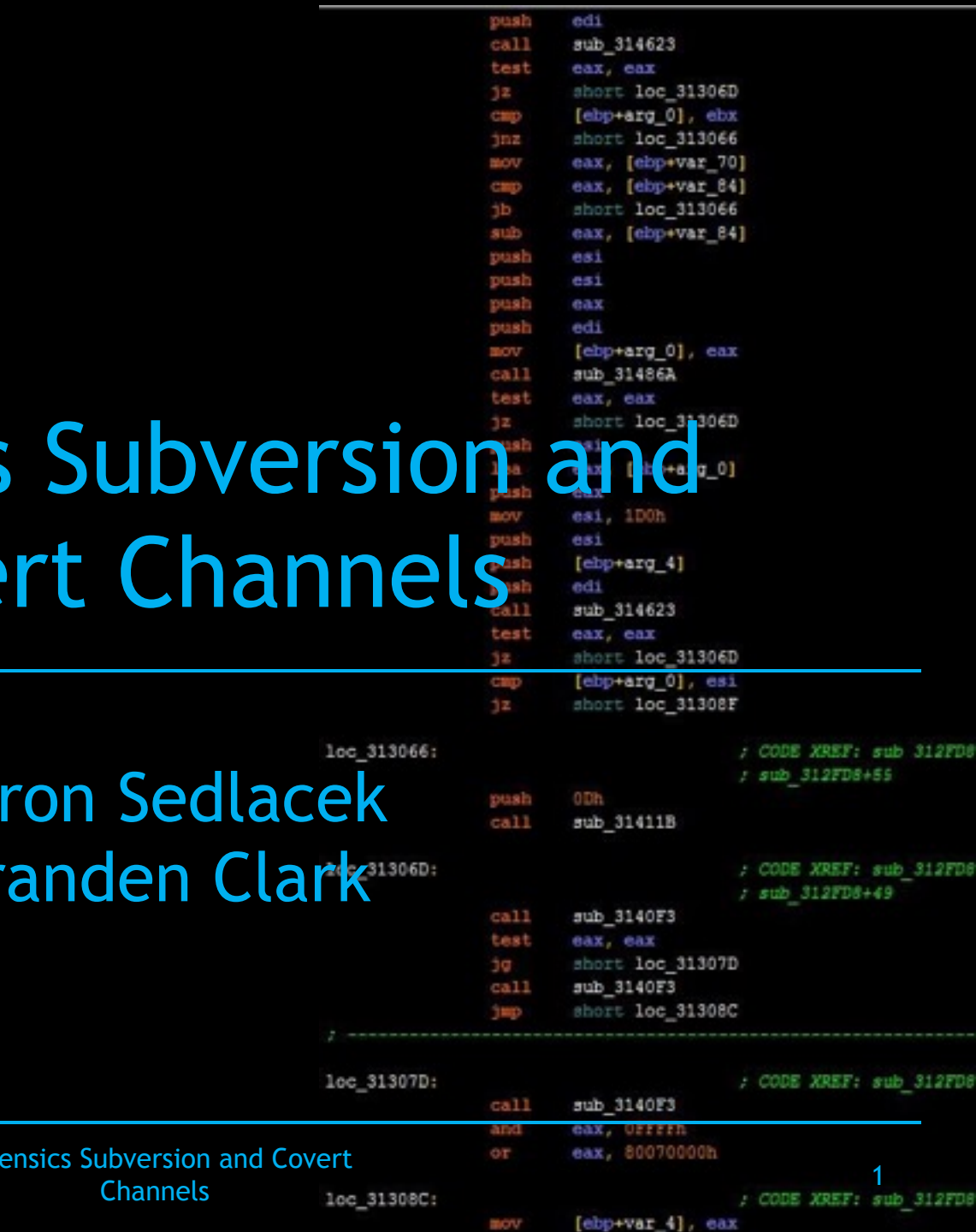
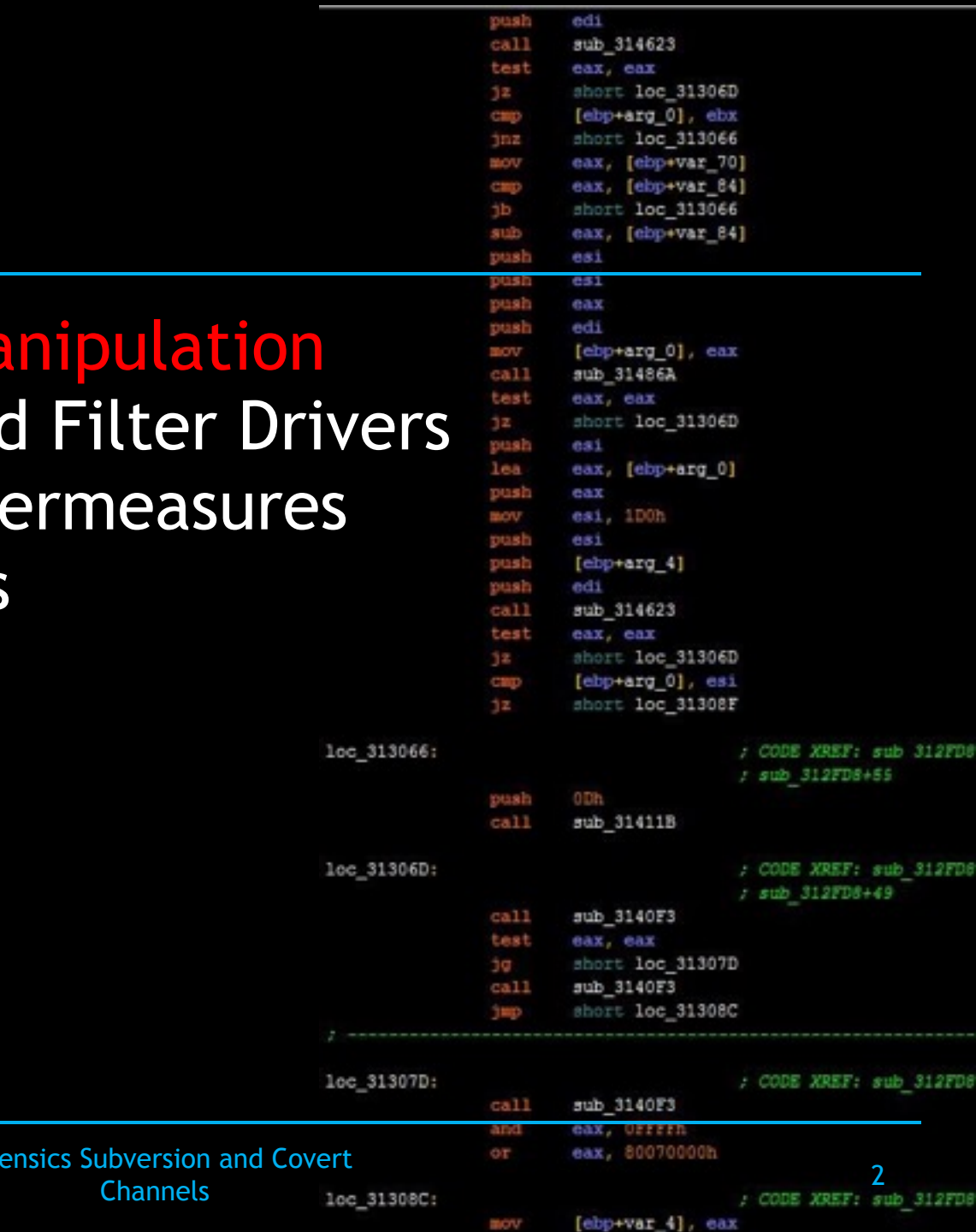
Forensics Subversion and Covert Channels

Aaron Sedlacek
Branden Clark



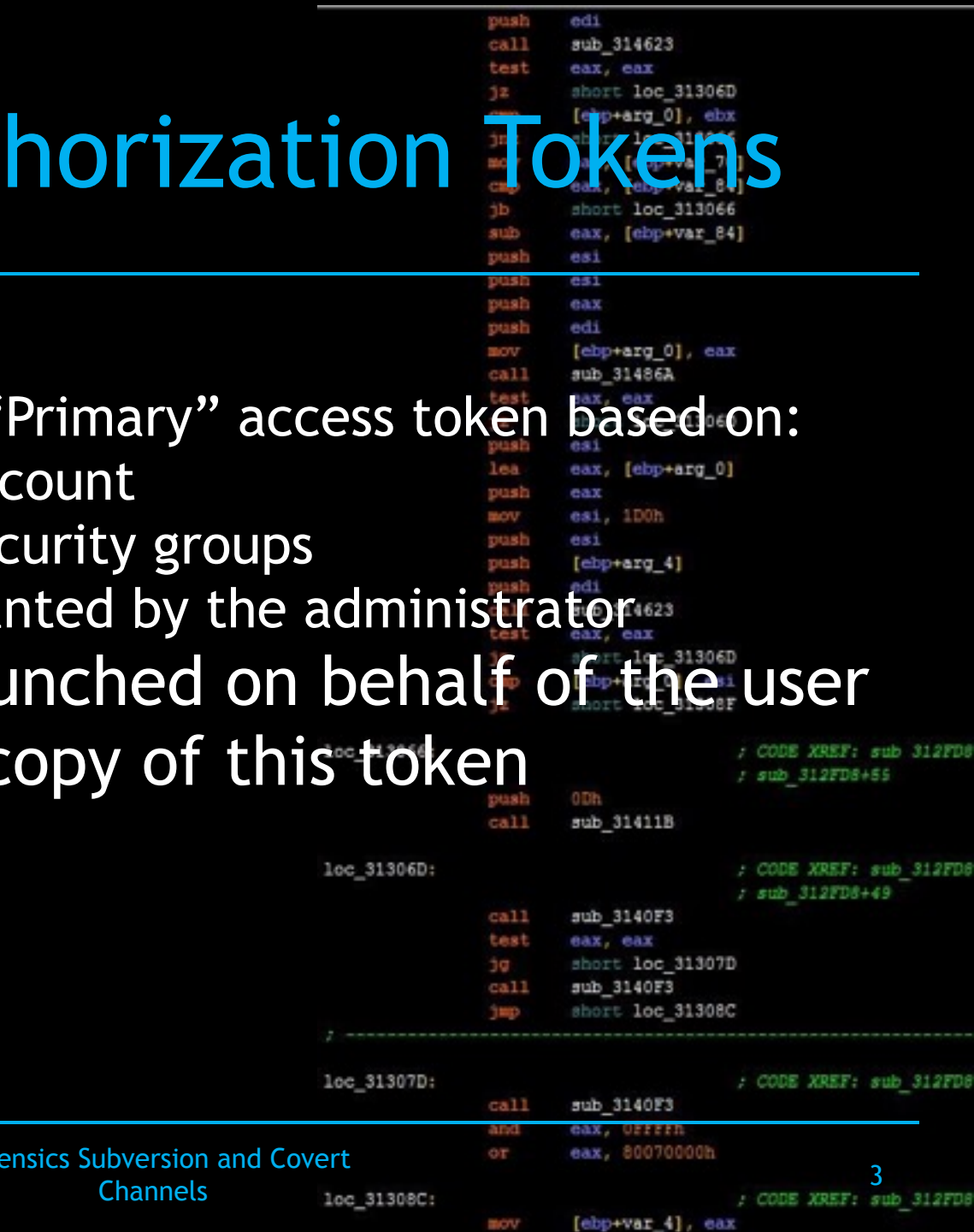
Agenda

- Access Token Manipulation
- IRP Handling and Filter Drivers
- Forensics Countermeasures
- Covert Channels
- Out-Of-Band



Windows Authorization Tokens

- User logs on
  - OS generates a “Primary” access token based on:
    - The user’s account
    - The user’s security groups
    - Privileges granted by the administrator
- All processes launched on behalf of the user will be given a copy of this token

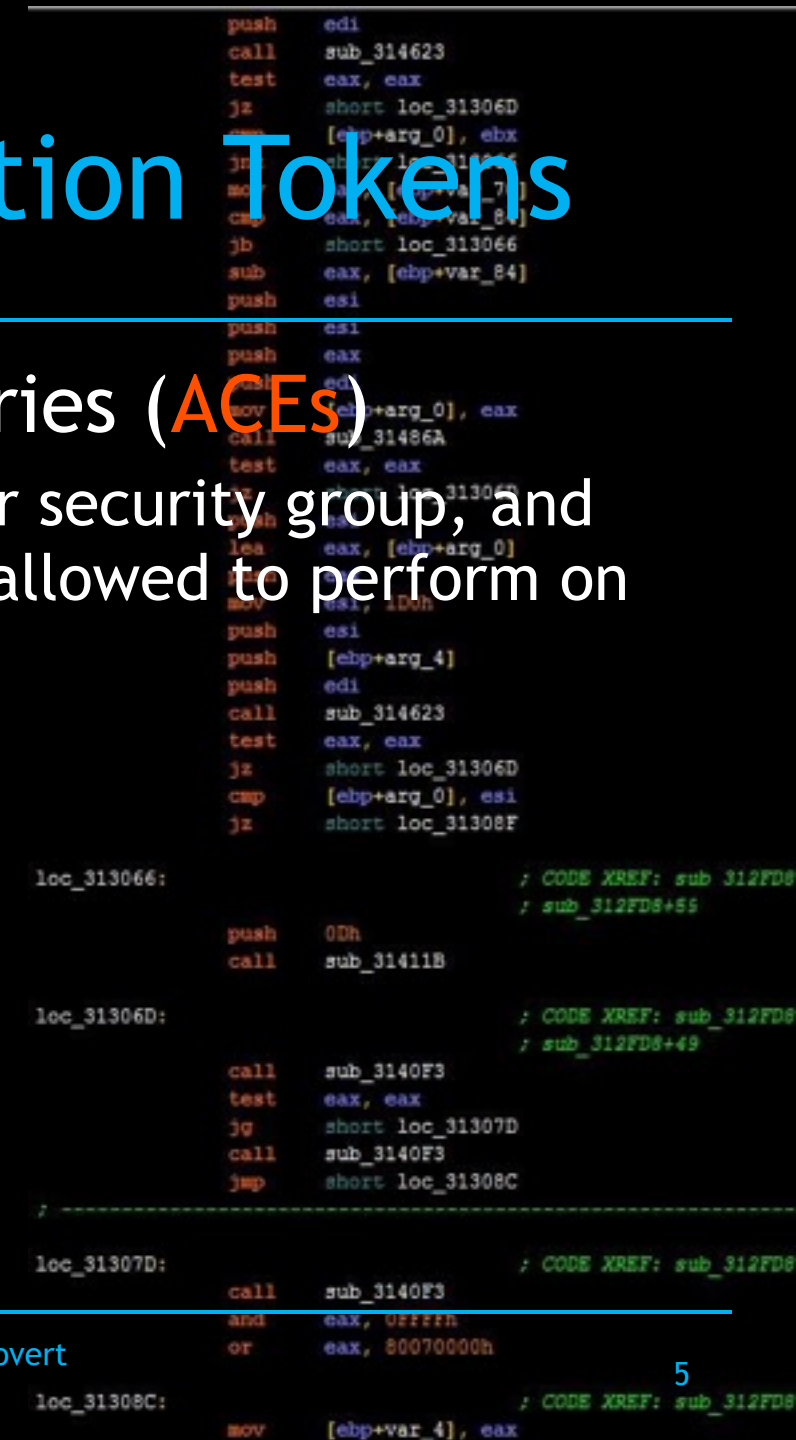


Windows Authorization Tokens

- The OS uses the access token to authorize a process when it attempts to
  - Perform an action that requires special privileges
  - Access a securable object
- Securable Object - Any object that has a security descriptor
- Security Descriptor
  - Determines an object's owner, primary security group, and discretionary access control list (DACL)

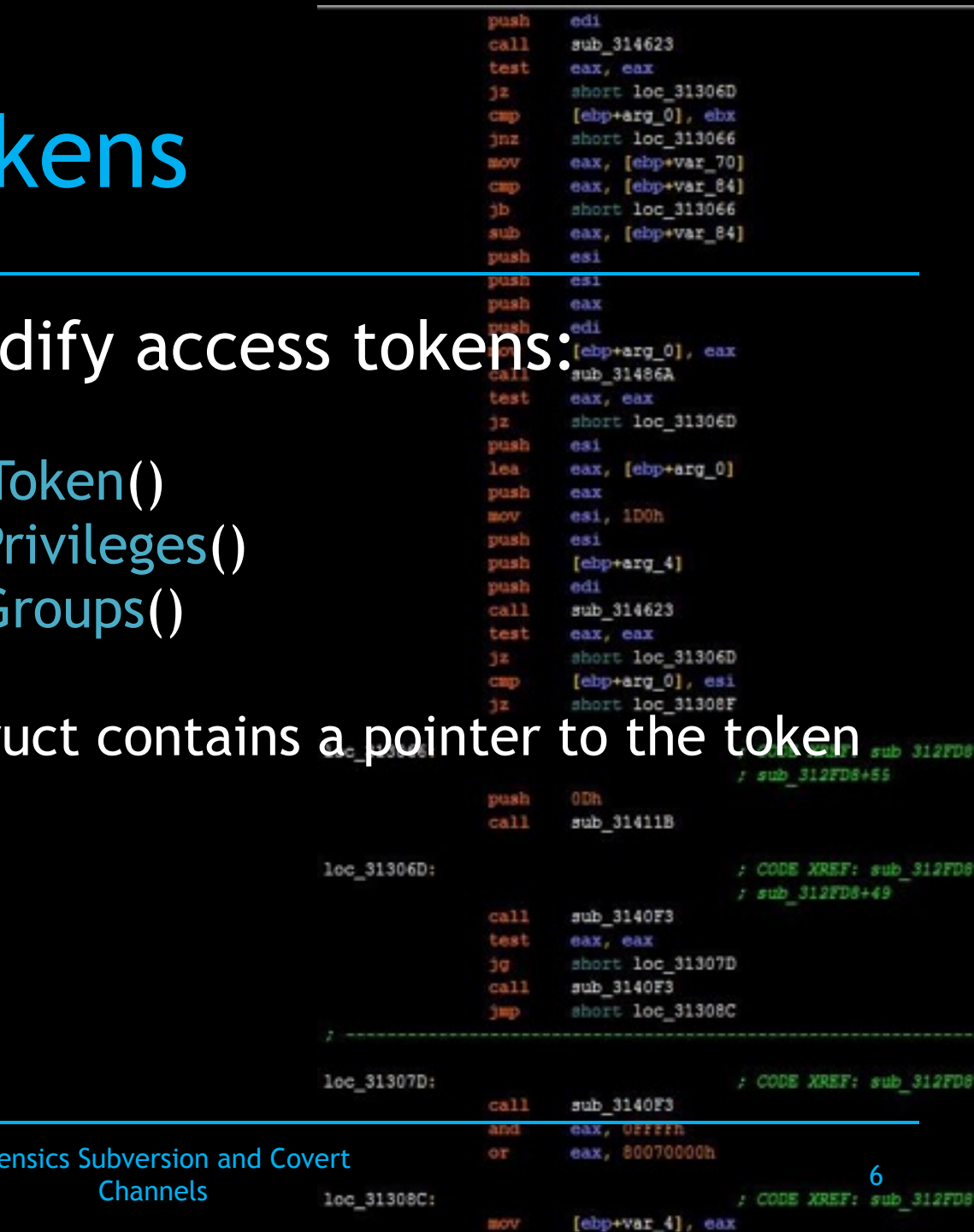
Windows Authorization Tokens

- List of access control entries (ACEs)
  - each ACE identifies a user, or security group, and the operations that they're allowed to perform on an object
- 34 different privileges
  - SeDebugPrivilege
  - SeShutdownPrivilege
  - SeLoadDriverPrivilege
  - ...



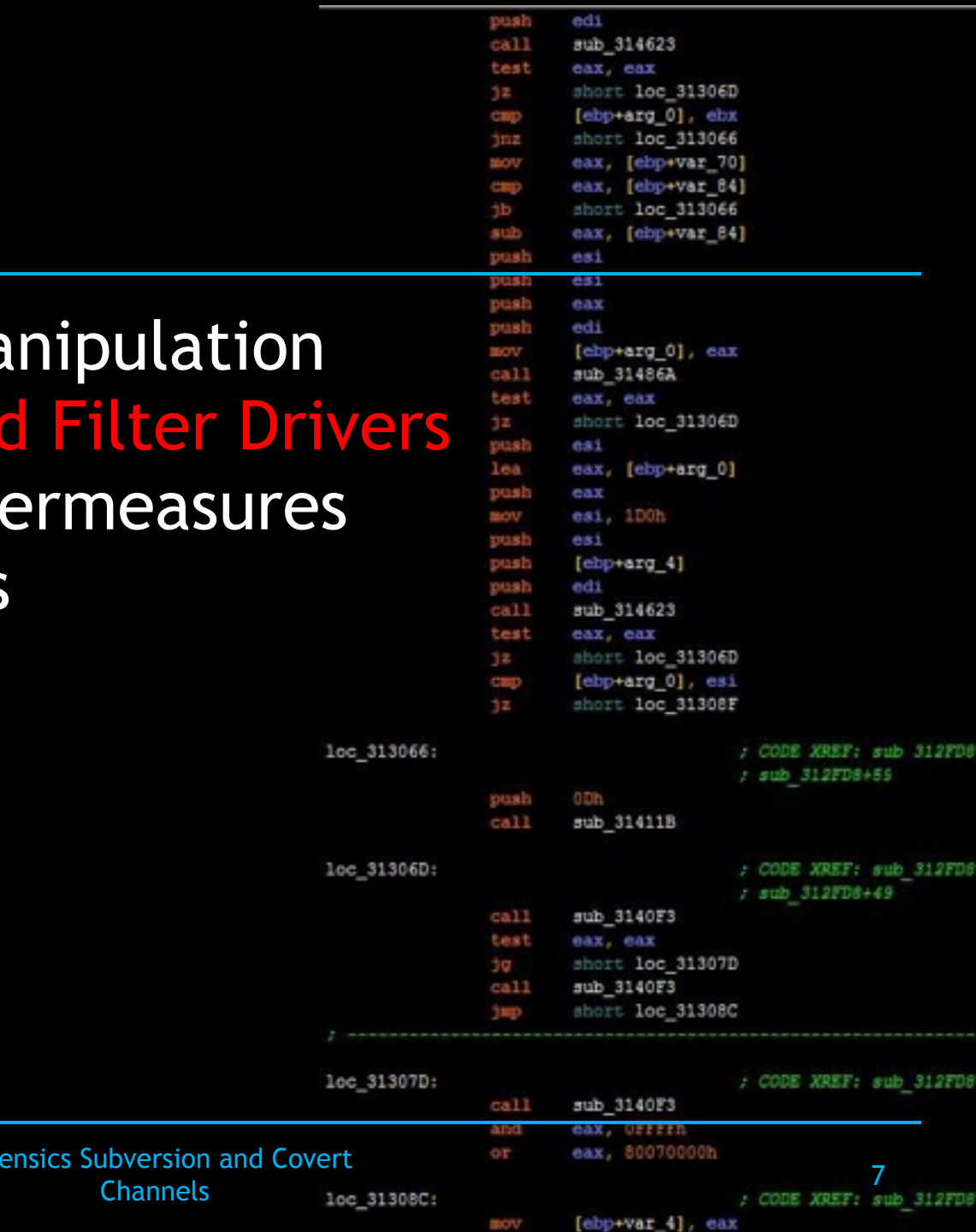
Modifying Tokens

- Two ways to modify access tokens:
  a. Windows API:
     - OpenProcessToken()
     - AdjustTokenPrivileges()
     - AdjustTokenGroups()
  b. DKOM
     - EPROCESS struct contains a pointer to the token object



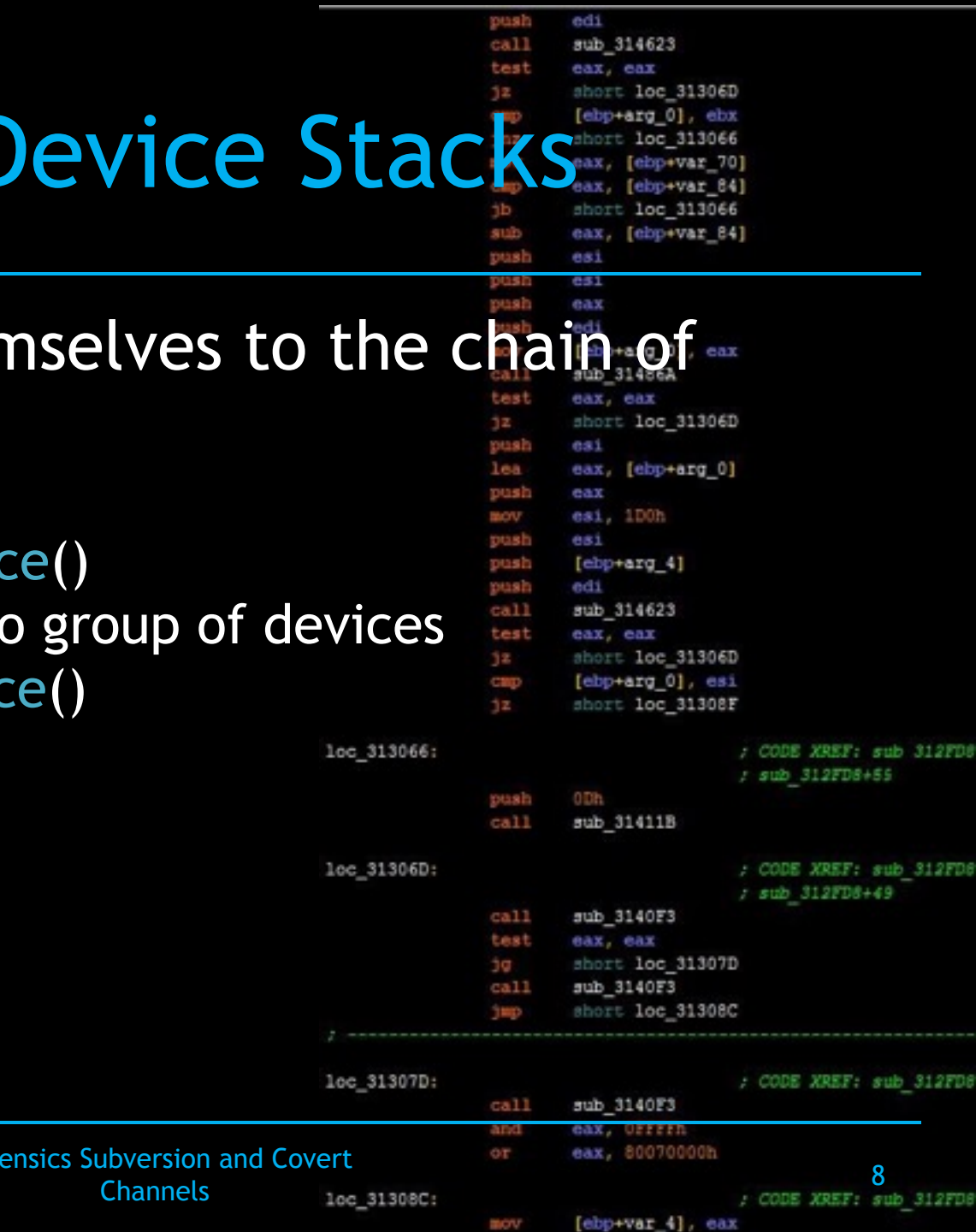
Agenda

- Access Token Manipulation
- IRP Handling and Filter Drivers
- Forensics Countermeasures
- Covert Channels
- Out-Of-Band

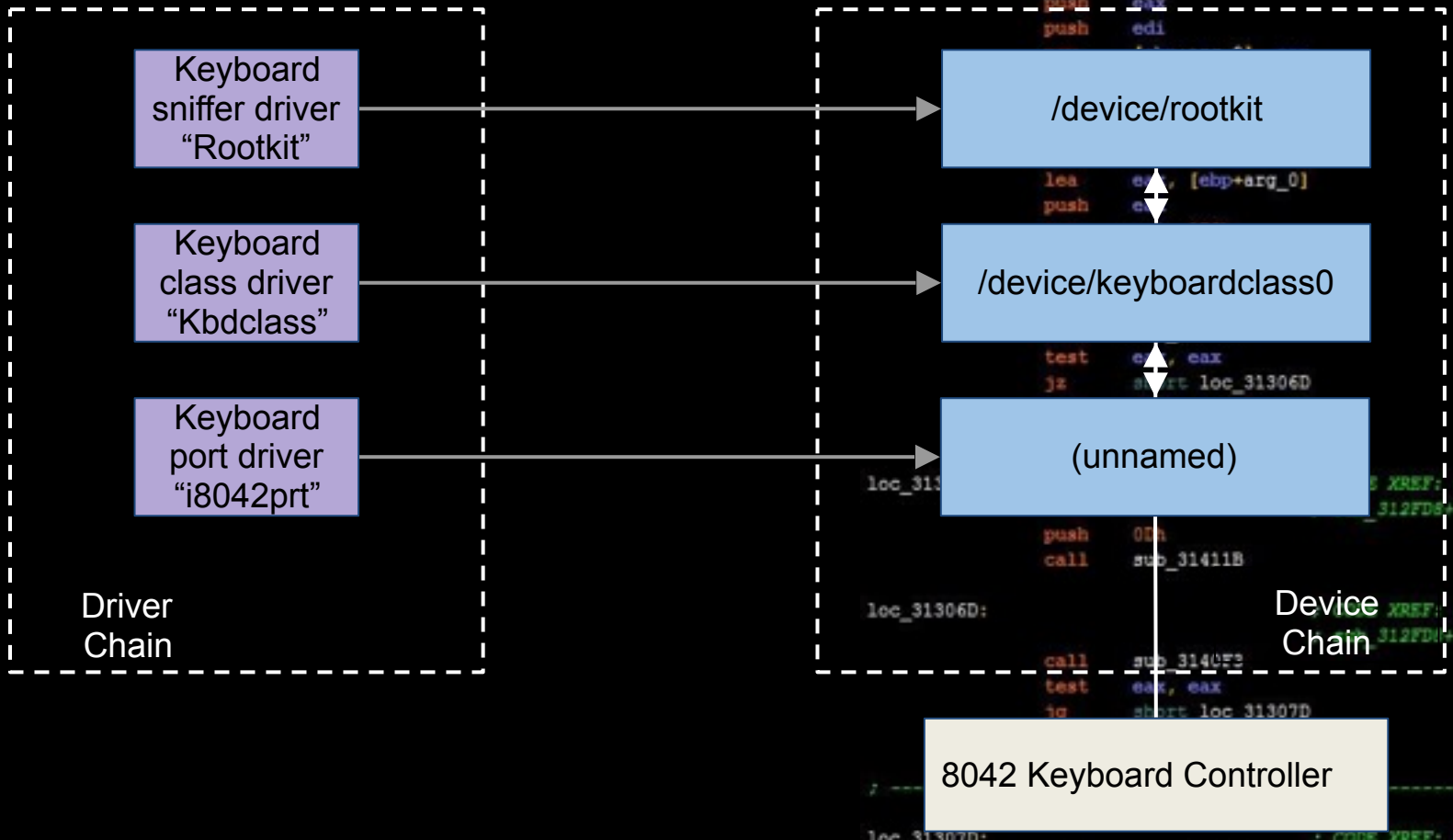


Drivers and Device Stacks

- Drivers add themselves to the chain of drivers
  - Create a device
    - IoCreateDevice()
  - Insert device into group of devices
    - IoAttachDevice()



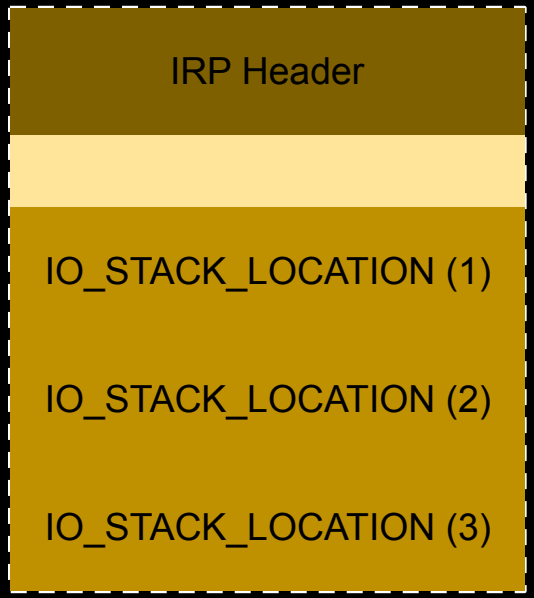
Drivers and Device Stacks

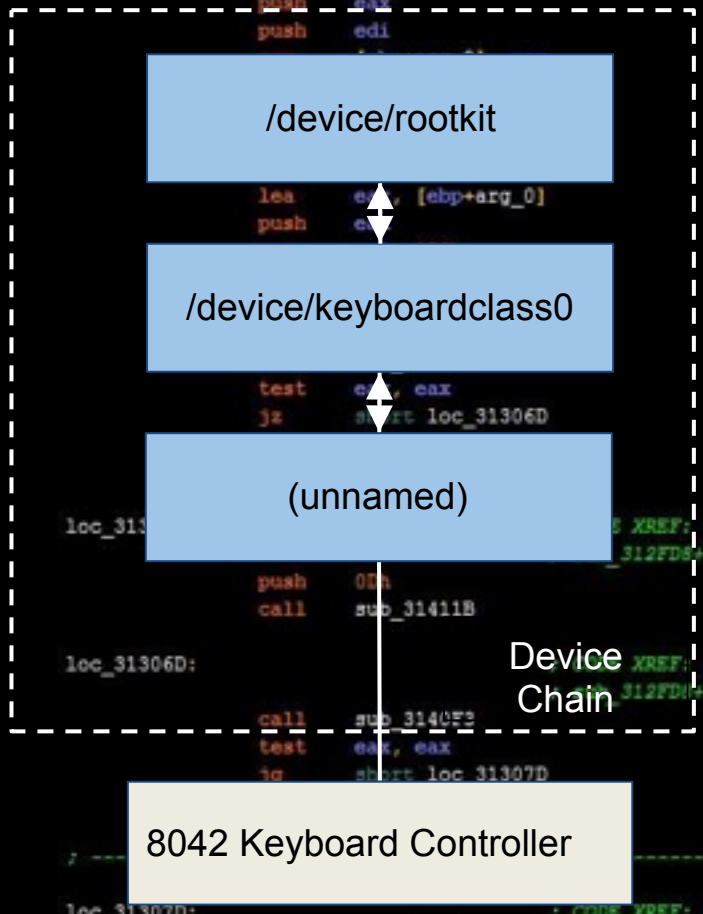


IRP Lifecycle Example

- Read request made to read keystroke

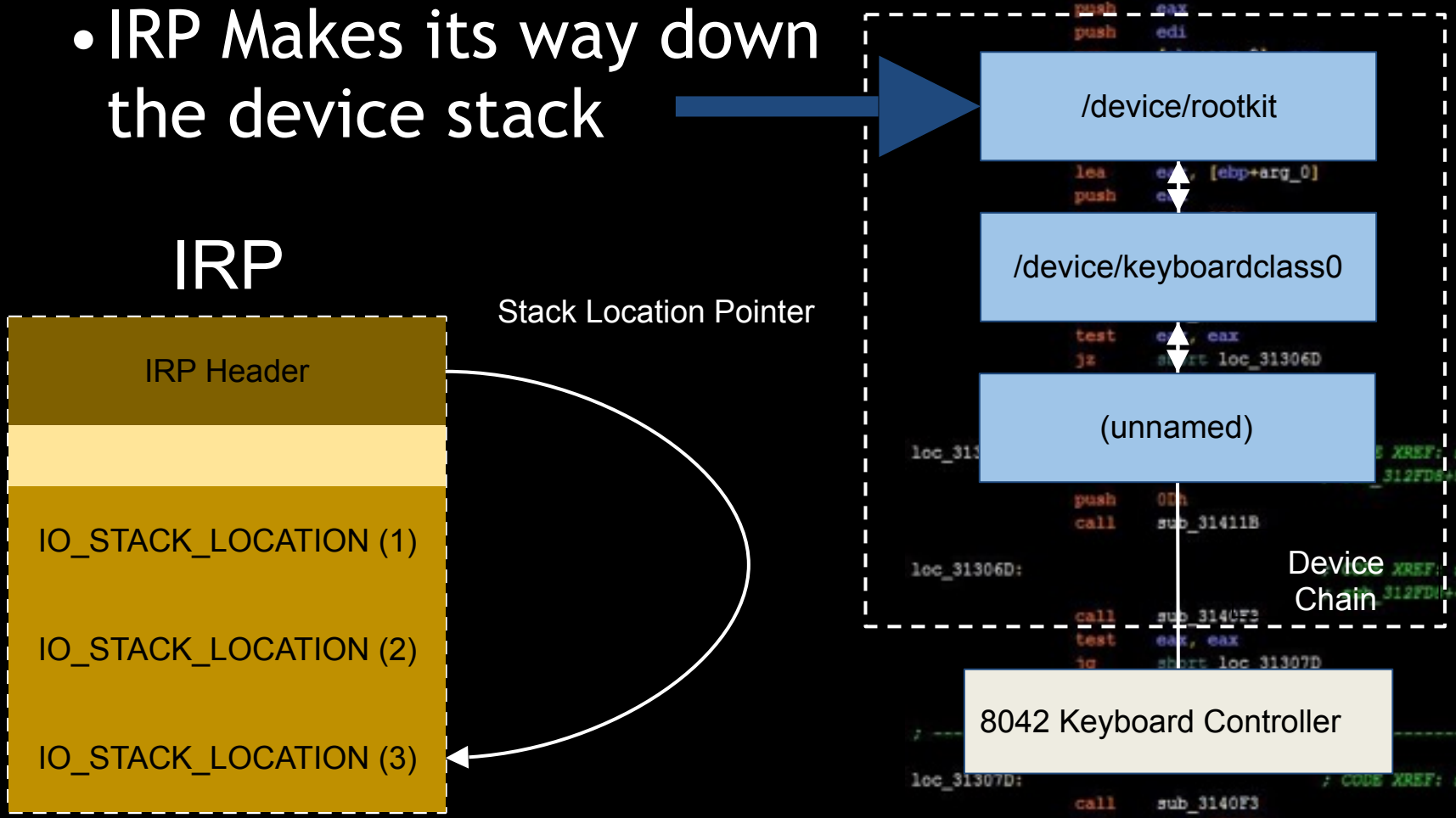
IRP



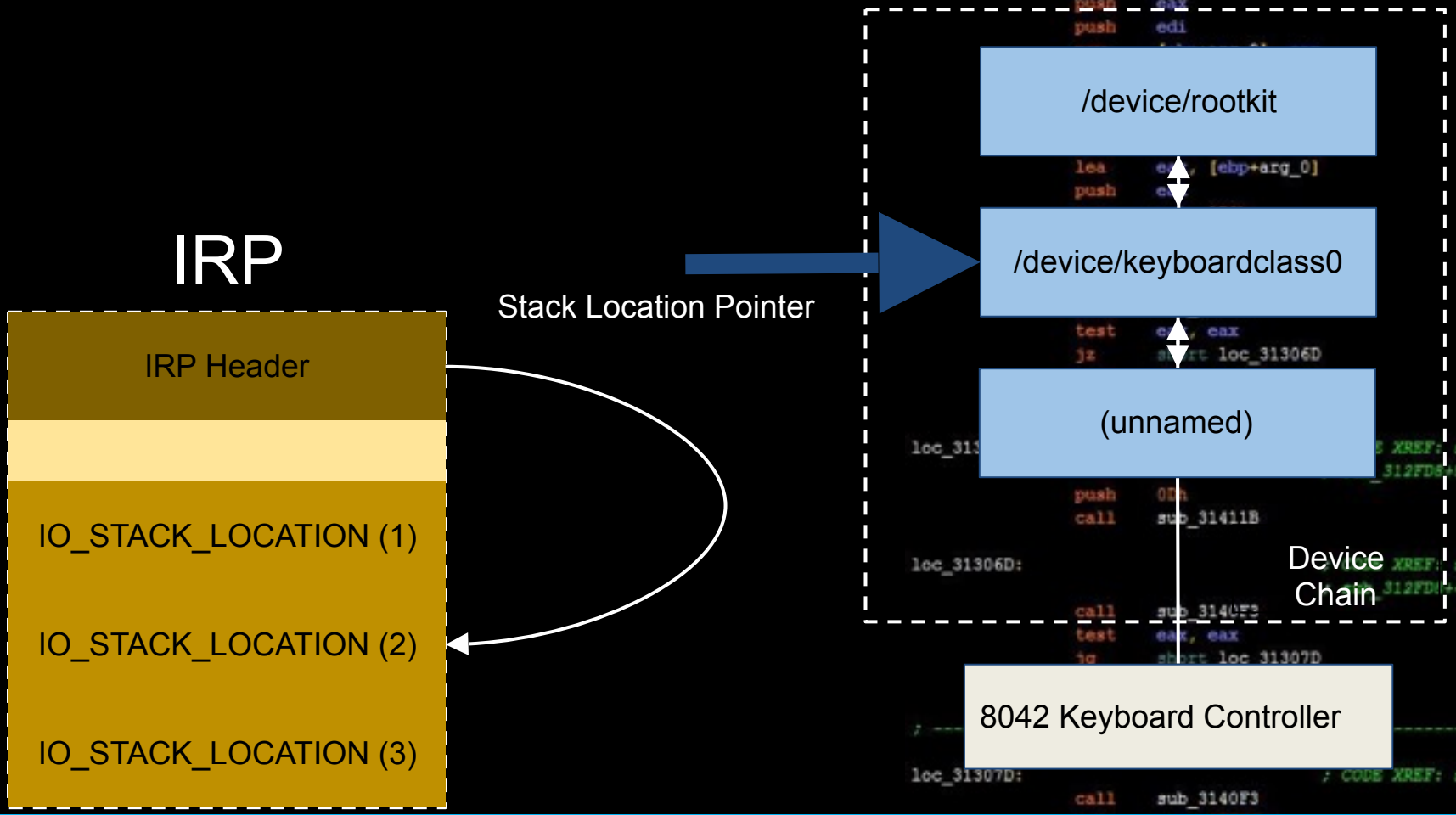


IRP Lifecycle Example

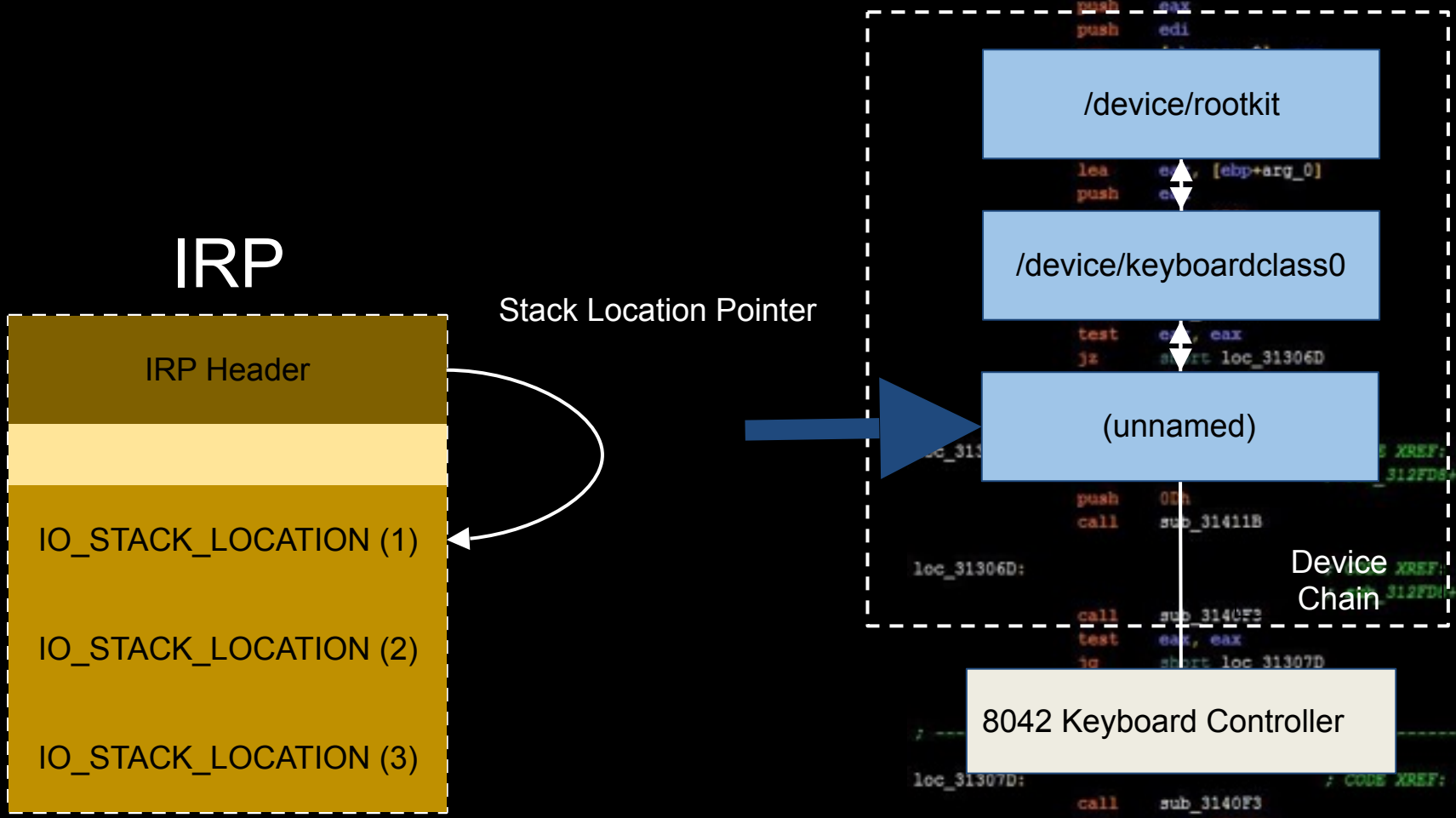
- IRP Makes its way down the device stack



IRP Lifecycle Example

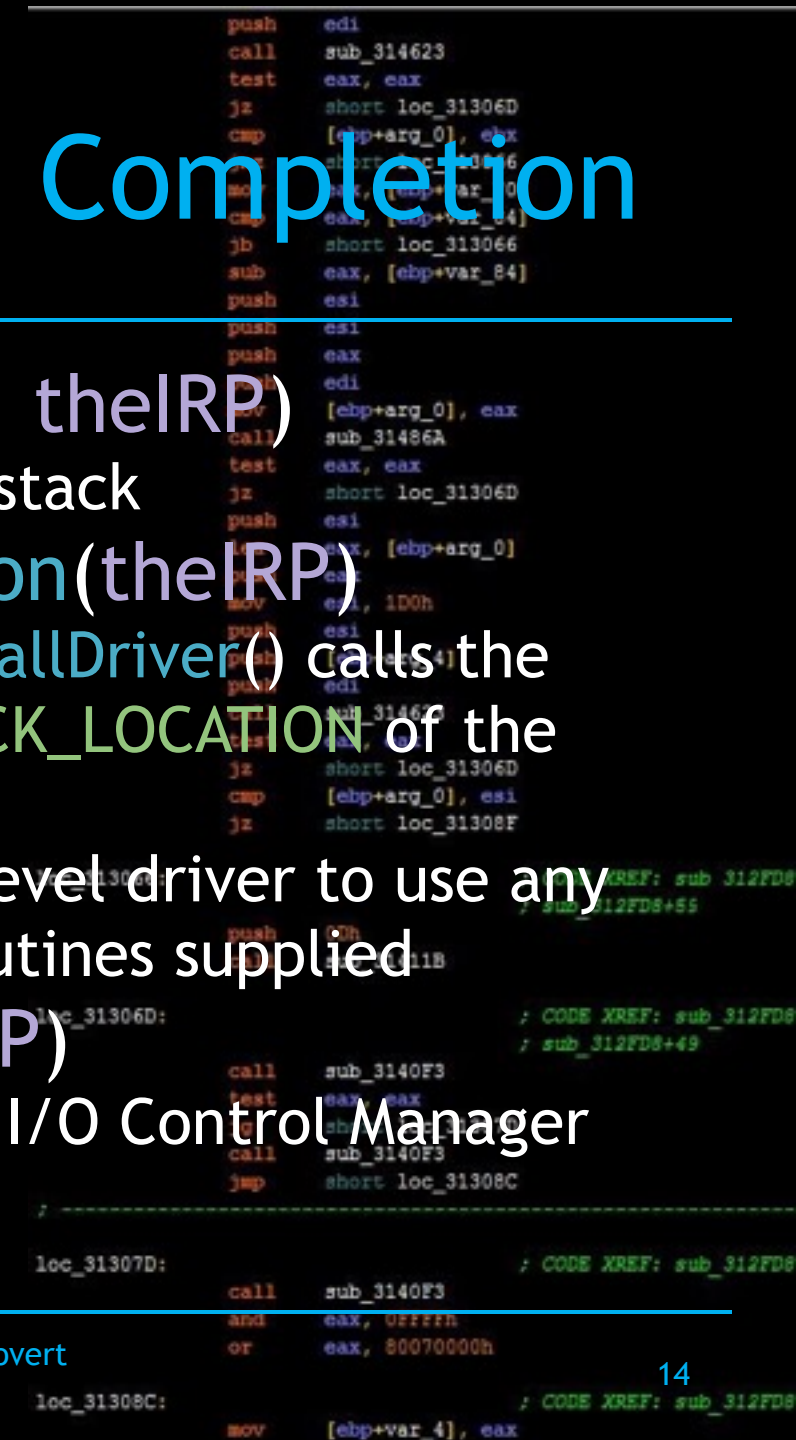


IRP Lifecycle Example



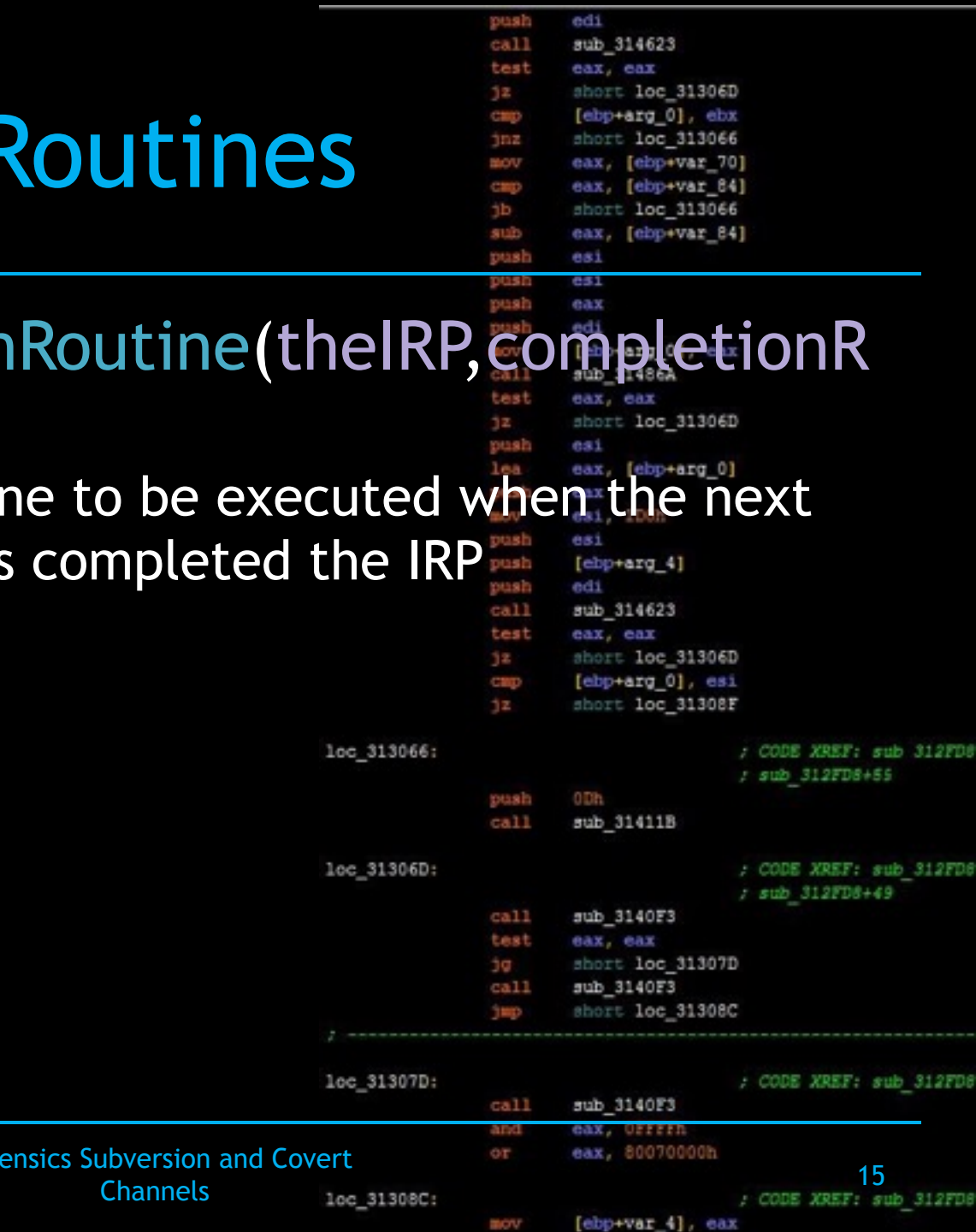
IRP Forwarding and Completion

- IoCallDriver(gNextDevice, theIRP)
  - Calls the next device in the stack
- IoSkipCurrentStackLocation(theIRP)
  - If this function is called, IoCallDriver() calls the next driver with the IO_STACK_LOCATION of the current driver
  - This trick allows the lower-level driver to use any arguments or completion routines supplied
- IoCompleteRequest(theIRP)
  - Returns the given IRP to the I/O Control Manager



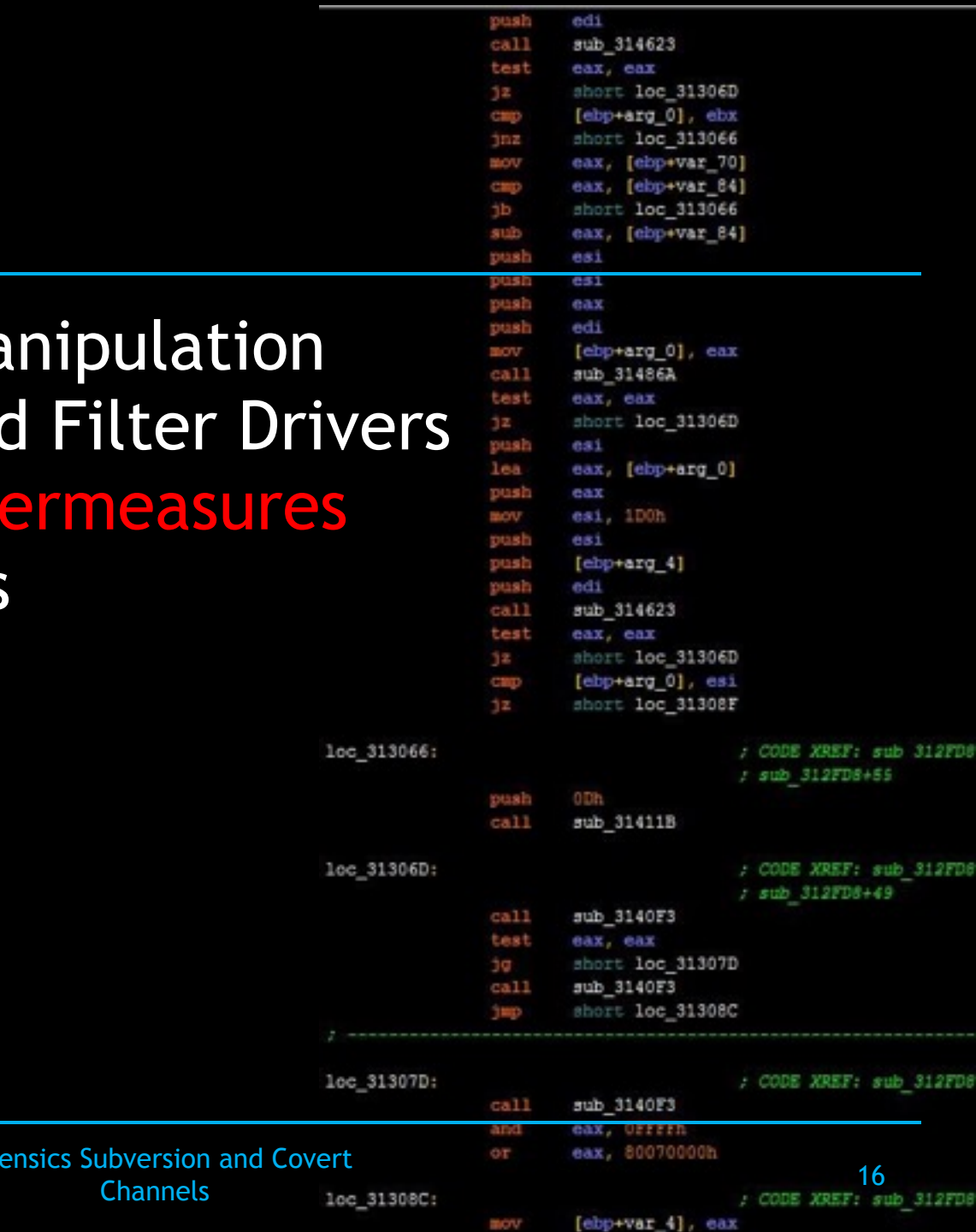
Completion Routines

- IoSetCompletionRoutine(theIRP,completionRoutine, ...)
  - Registers a routine to be executed when the next lowest driver has completed the IRP



Agenda

- Access Token Manipulation
- IRP Handling and Filter Drivers
- Forensics Countermeasures
- Covert Channels
- Out-Of-Band



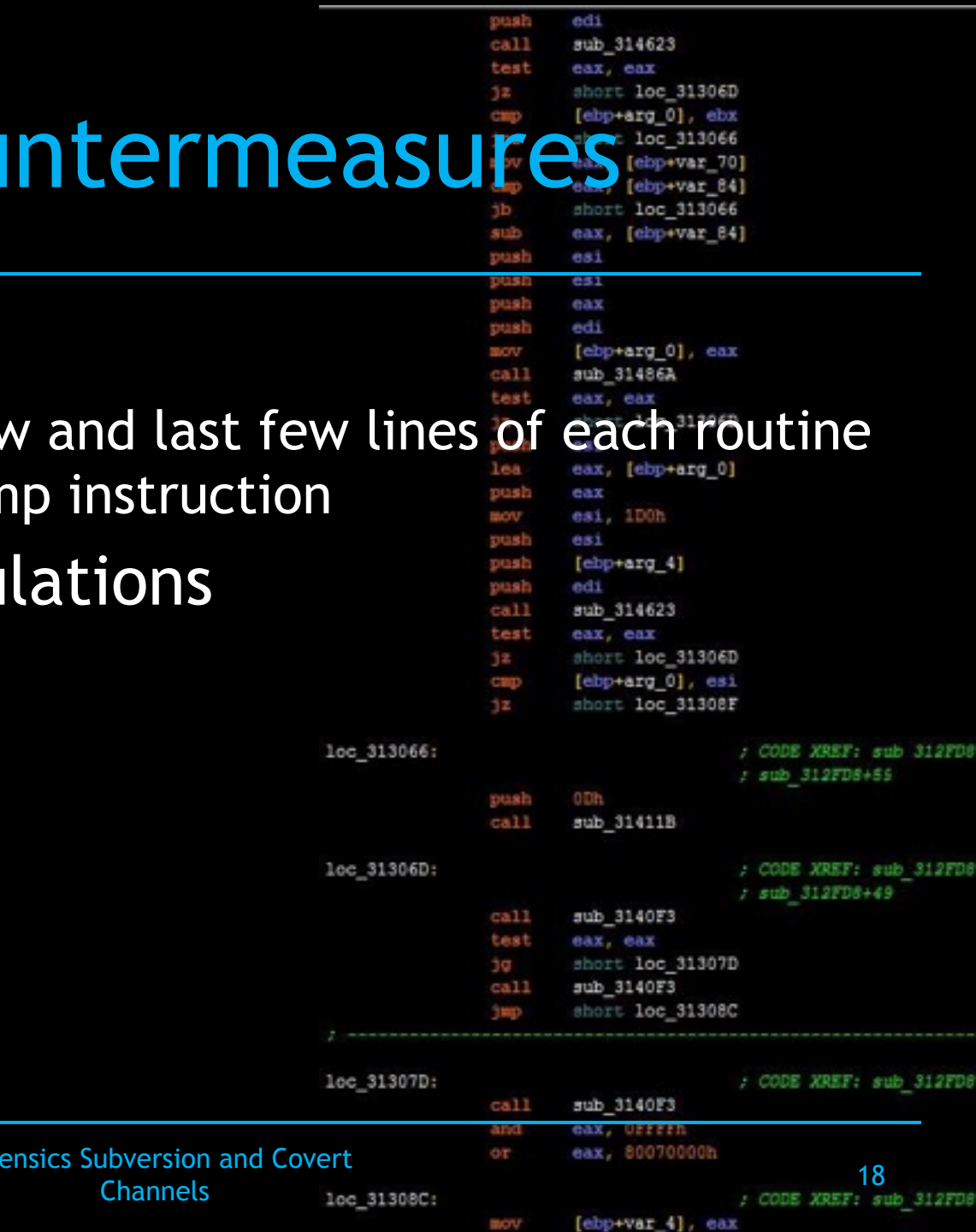
Hooking Countermeasures

- There are certain ranges of addresses that most call table entries should contain

Call Table	Red Flag Condition
IAT	An entry lies outside of its designated DLL address range
IDT	The address of the 0x2E handler lies outside of ntoskrnl.exe
MSR	The contents of IA32_SYSENTER_EIP lie outside of ntoskrnl.exe
SSDT	Pointers to the Nt*() routines lie outside of ntoskrnl.exe
IRP	The addresses of dispatch routines lie outside of the driver's address range

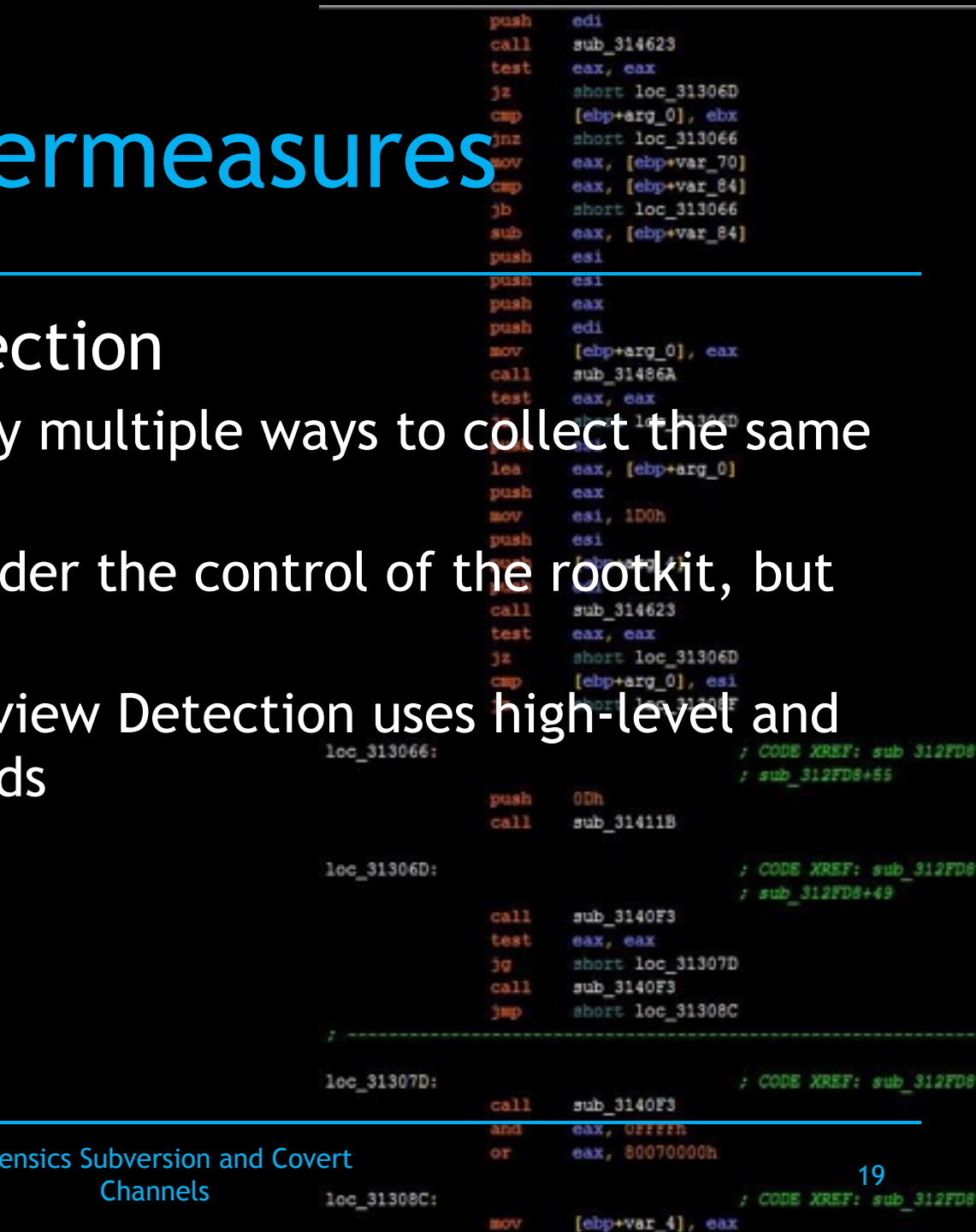
Patching Countermeasures

- Byte Scanning
  - Scan the first few and last few lines of each routine for a telltale jump instruction
- Checksum Calculations



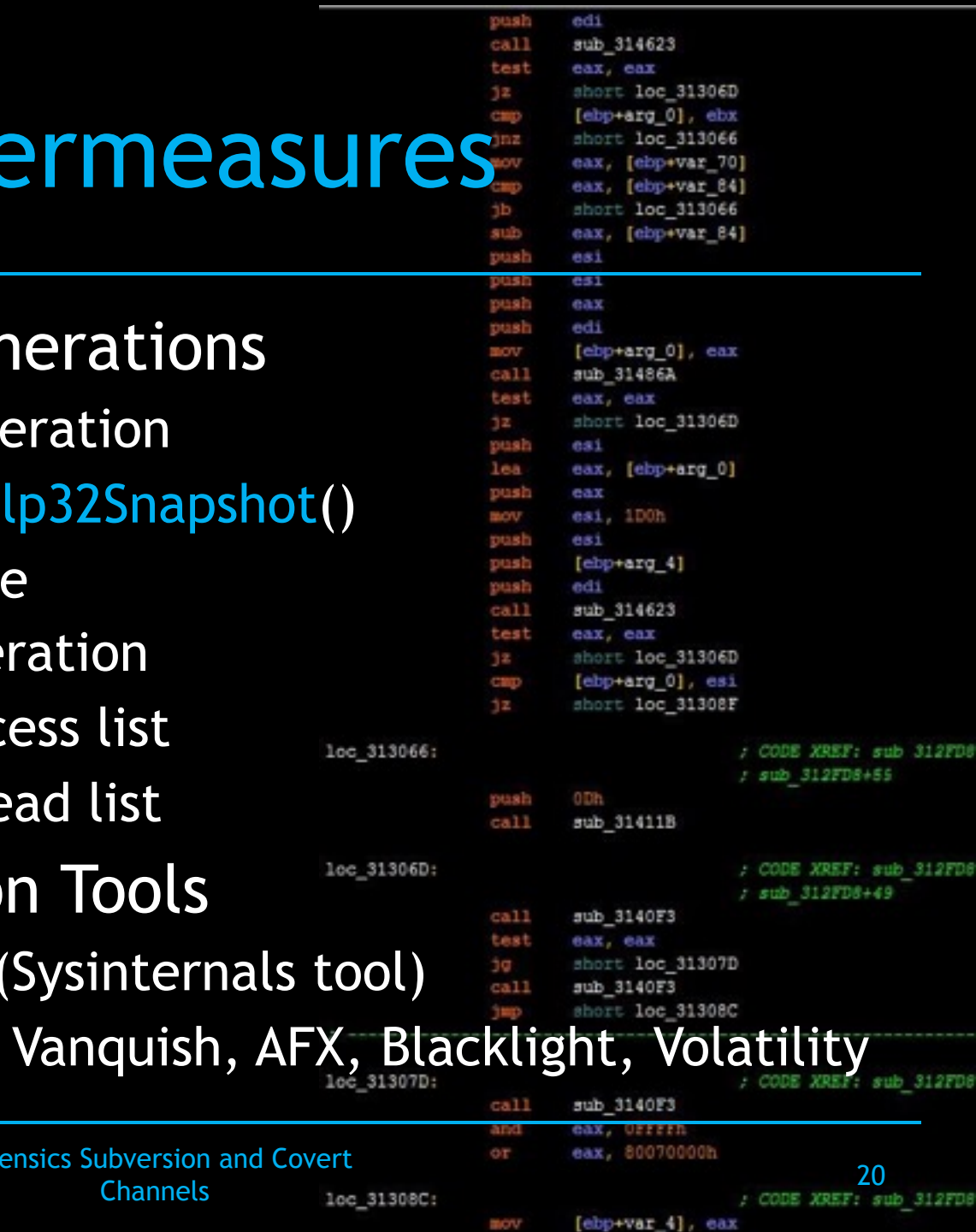
DKOM Countermeasures

- Cross-View Detection
  - There are usually multiple ways to collect the same information
  - Some may be under the control of the rootkit, but not all
  - Typically, Cross-view Detection uses high-level and low-level methods



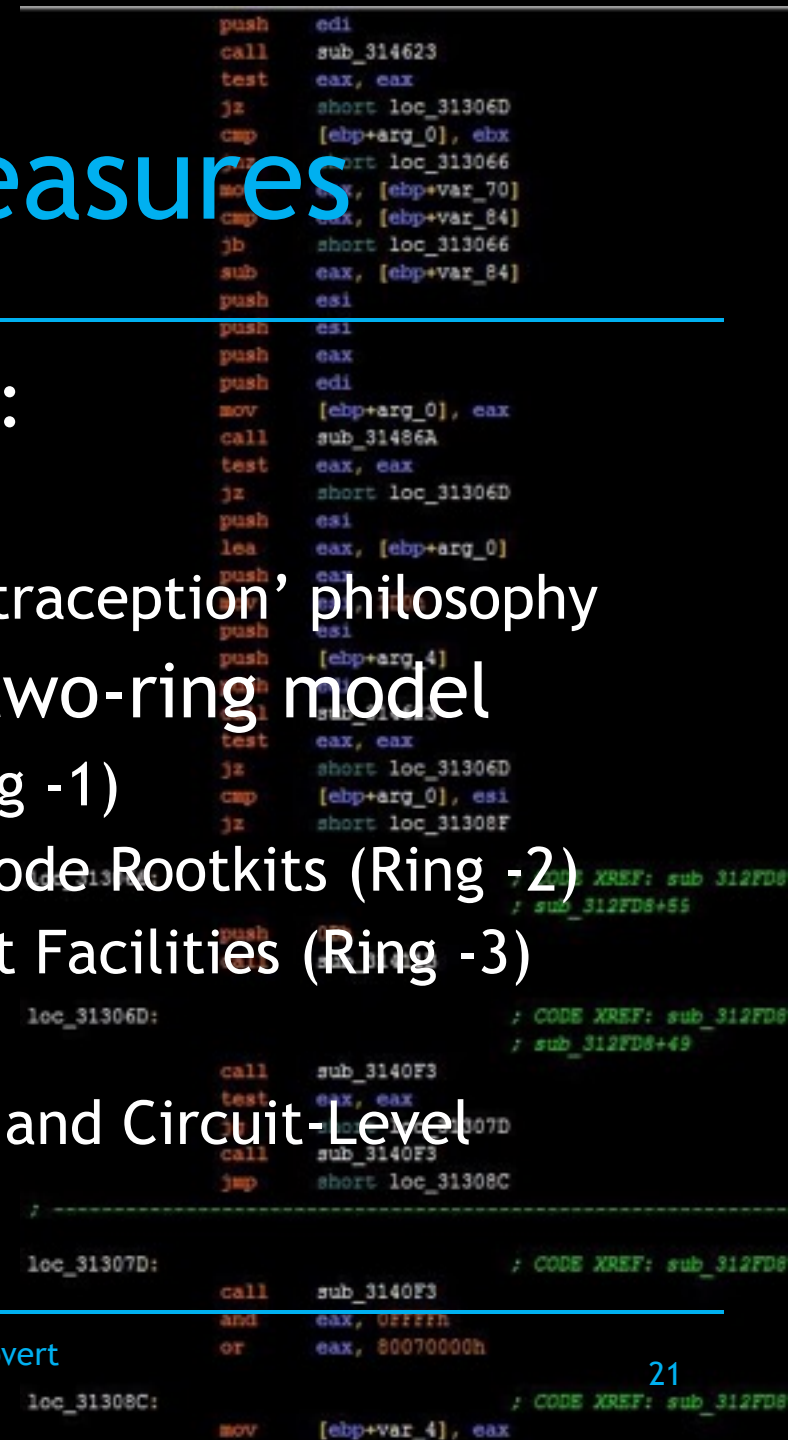
DKOM Countermeasures

- Cross-View Enumerations
  - High-Level Enumeration
    - CreateToolHelp32Snapshot()
    - PID Bruteforce
  - Low-level Enumeration
    - Walk the Process list
    - Walk the Thread list
- Rootkit Detection Tools
  - RootkitRevealer (Sysinternals tool)
  - HackerDefender, Vanquish, AFX, Blacklight, Volatility



Counter-Countermeasures

- Authors have three options:
  1. Starve the Opposition
     a. The Grugq's 'Data Contraception' philosophy
  2. Stepping out of the two-ring model
     a. Rogue Hypervisors (Ring -1)
     b. System Management Mode Rootkits (Ring -2)
     c. Lights-Out Management Facilities (Ring -3)
     d. Bootkits
     e. Onboard Flash Storage and Circuit-Level
  3. Covert Channels

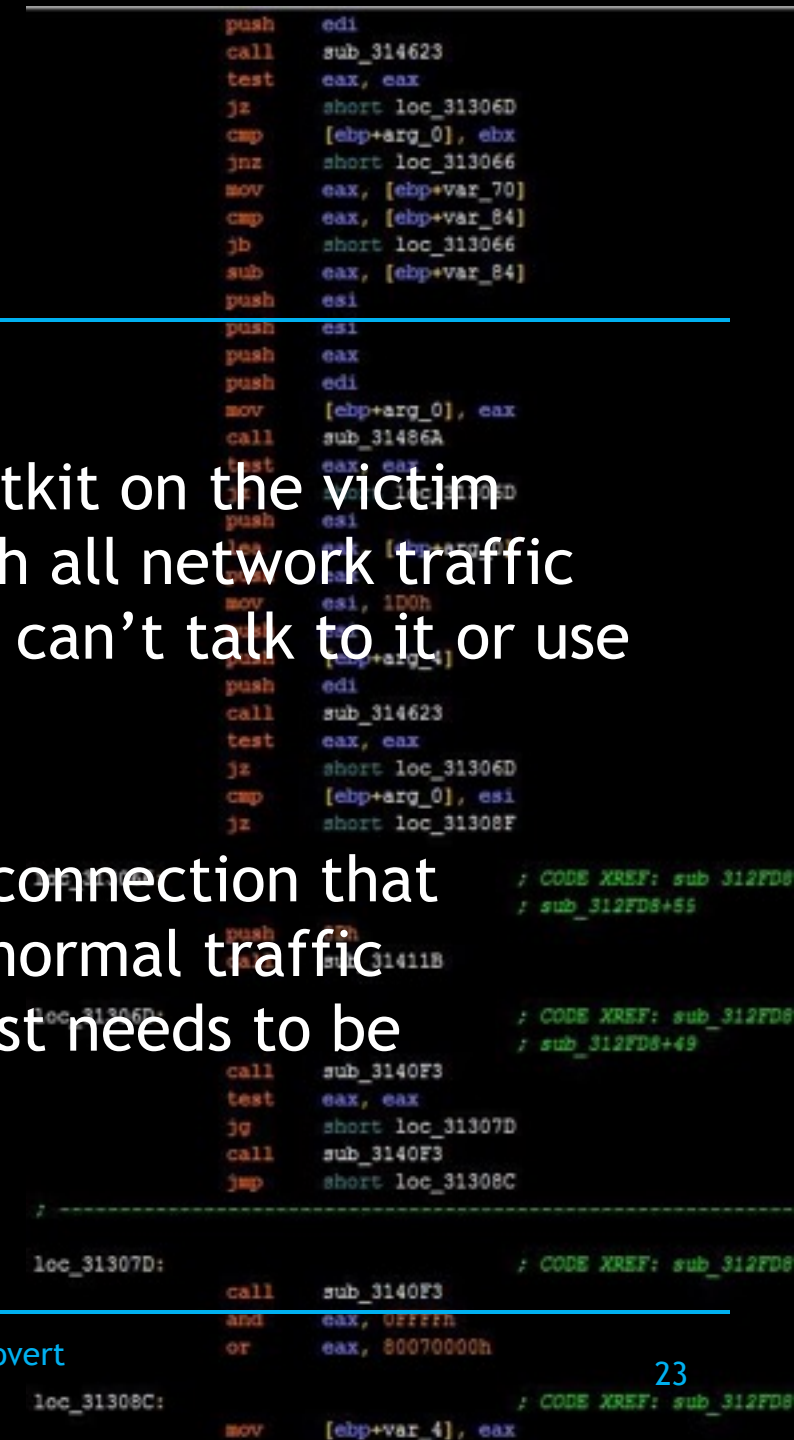


Agenda

- Access Token Manipulation
- IRP Handling and Filter Drivers
- Forensics Countermeasures
- Covert Channels
- Out-Of-Band

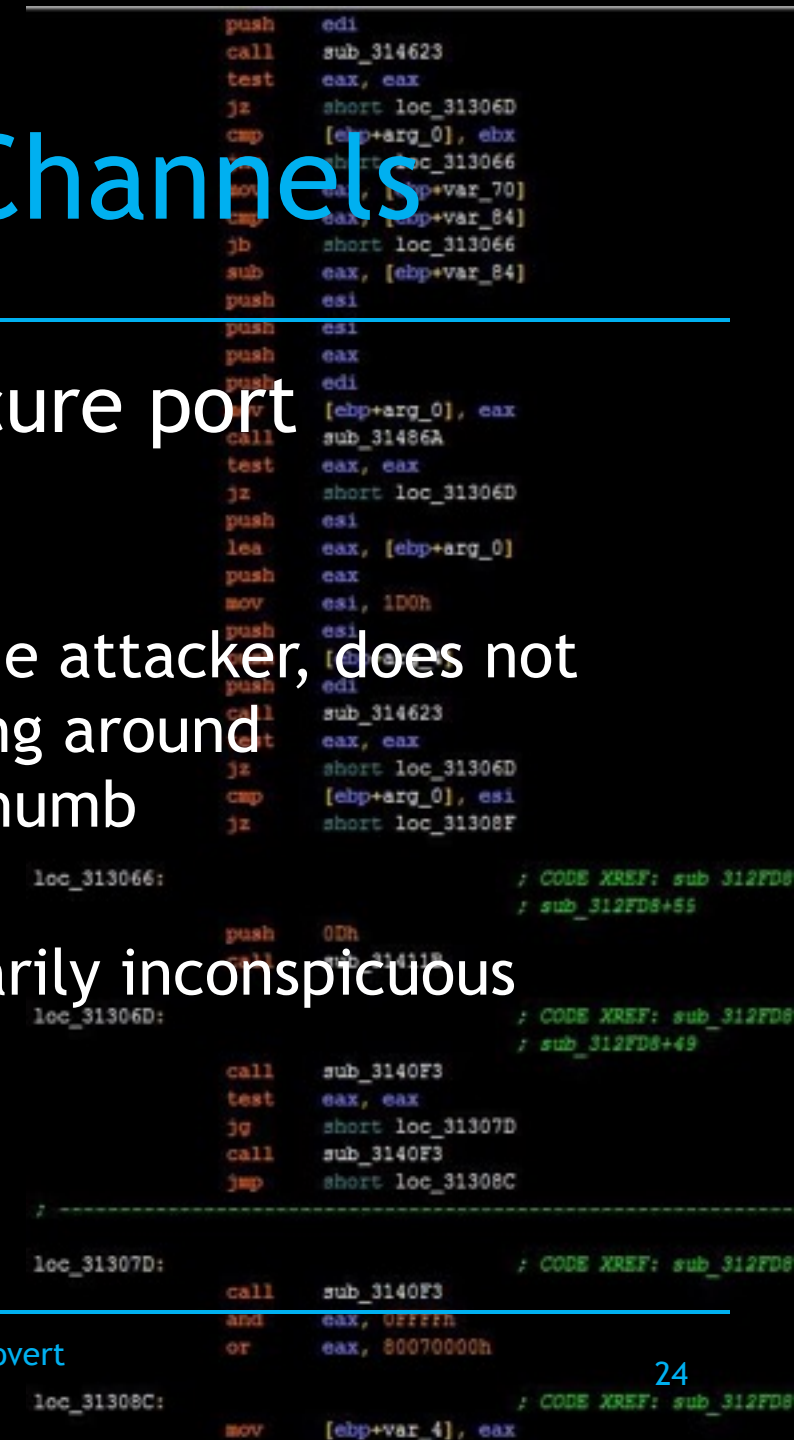
Covert Channels

- Why?
  - If an analyst can't find a rootkit on the victim machine, they can still watch all network traffic
  - What good is a rootkit if you can't talk to it or use it to retrieve data?
- How?
  - Covert Channel - A network connection that disguises its byte stream as normal traffic
  - Doesn't need to be fancy, just needs to be unanticipated



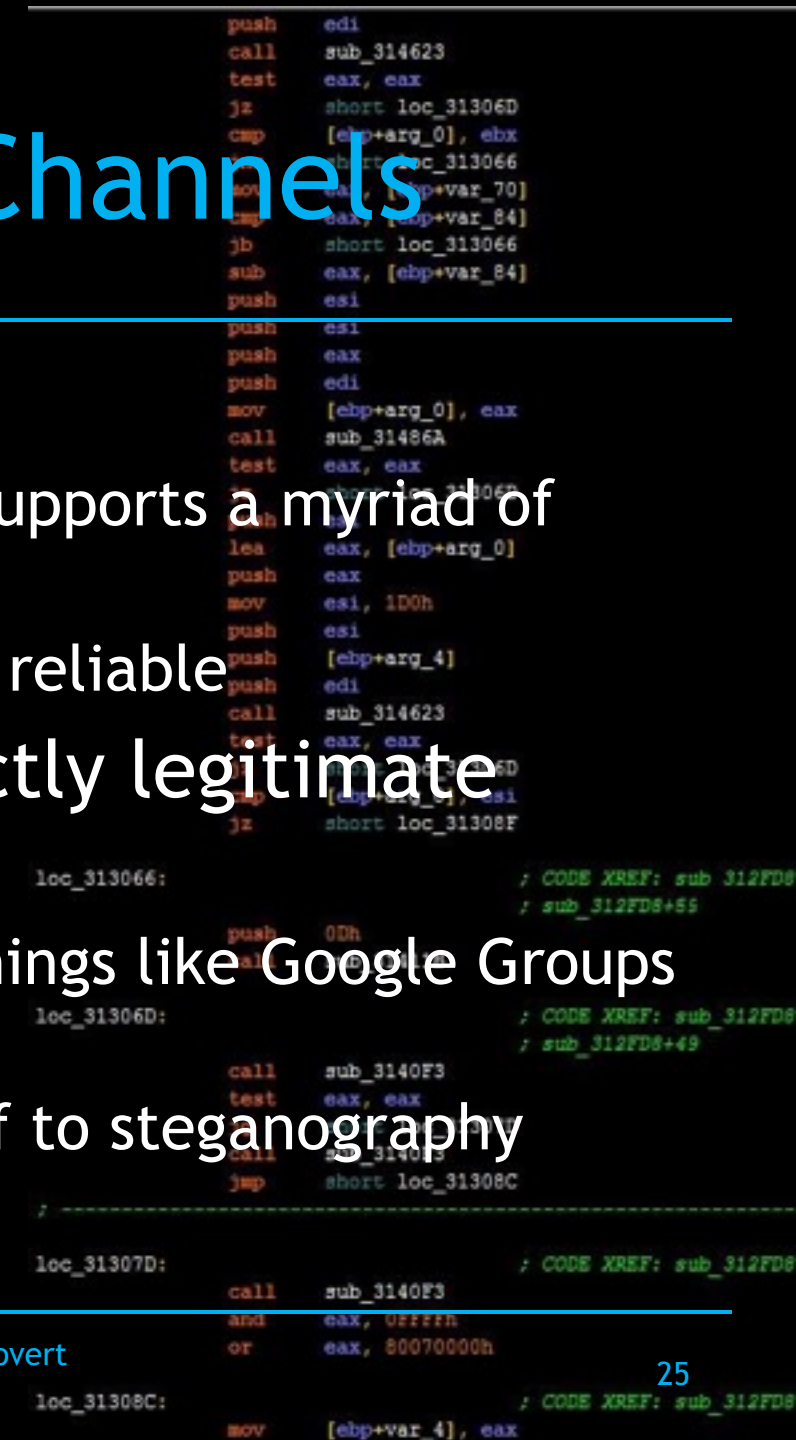
Common Malware Channels

- 1980's - Listen on an obscure port
  - Firewalls
- Late 1980's - IRC
  - Malware connects back to the attacker, does not leave an obscure port hanging around
  - Now stands out like a sore thumb
- 1990's - Peer-to-peer
  - P2P protocols aren't necessarily inconspicuous



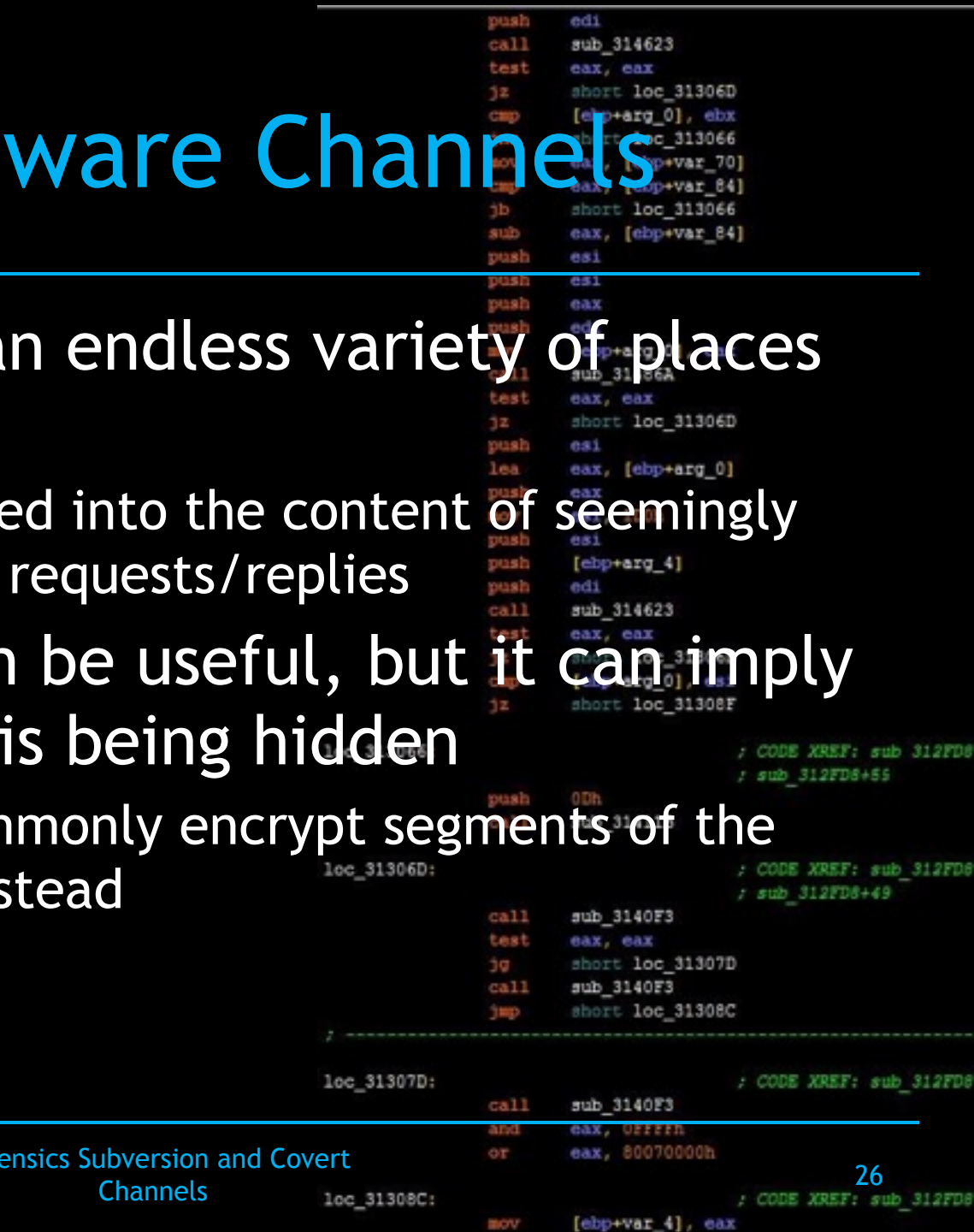
Common Malware Channels

- Now - HTTP
  - Web traffic is everywhere, supports a myriad of higher-level protocols
  - with TCP, can be considered reliable
- Malware will stick to strictly legitimate URLs
  - Malware has started using things like Google Groups as C2 servers
  - Difficult to filter, lends itself to steganography



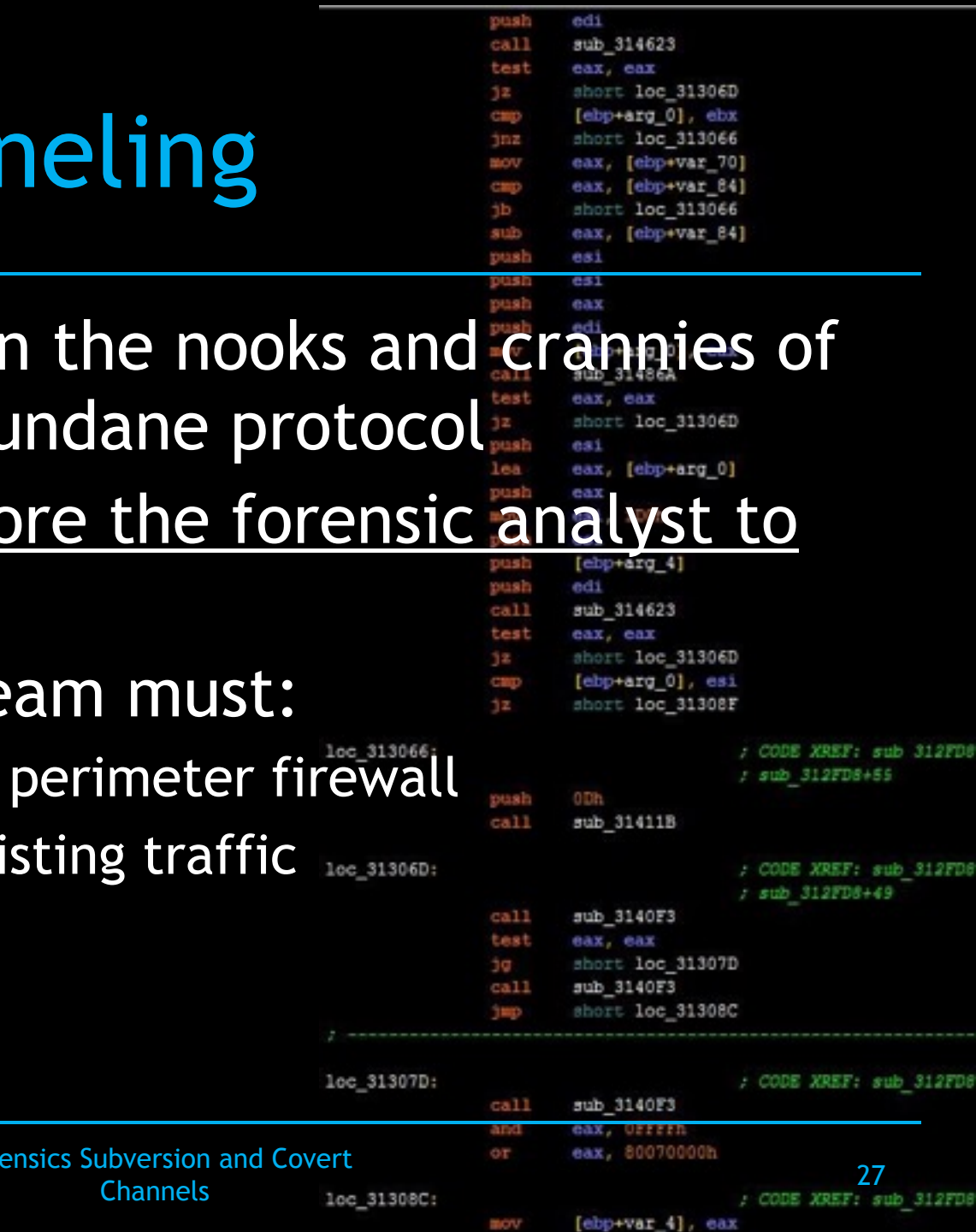
Common Malware Channels

- HTTP provides an endless variety of places to hide data
  - Particularly, mixed into the content of seemingly legitimate HTTP requests/replies
  - Using HTTPS can be useful, but it can imply that some data is being hidden
  - Rootkits will commonly encrypt segments of the HTTP content instead



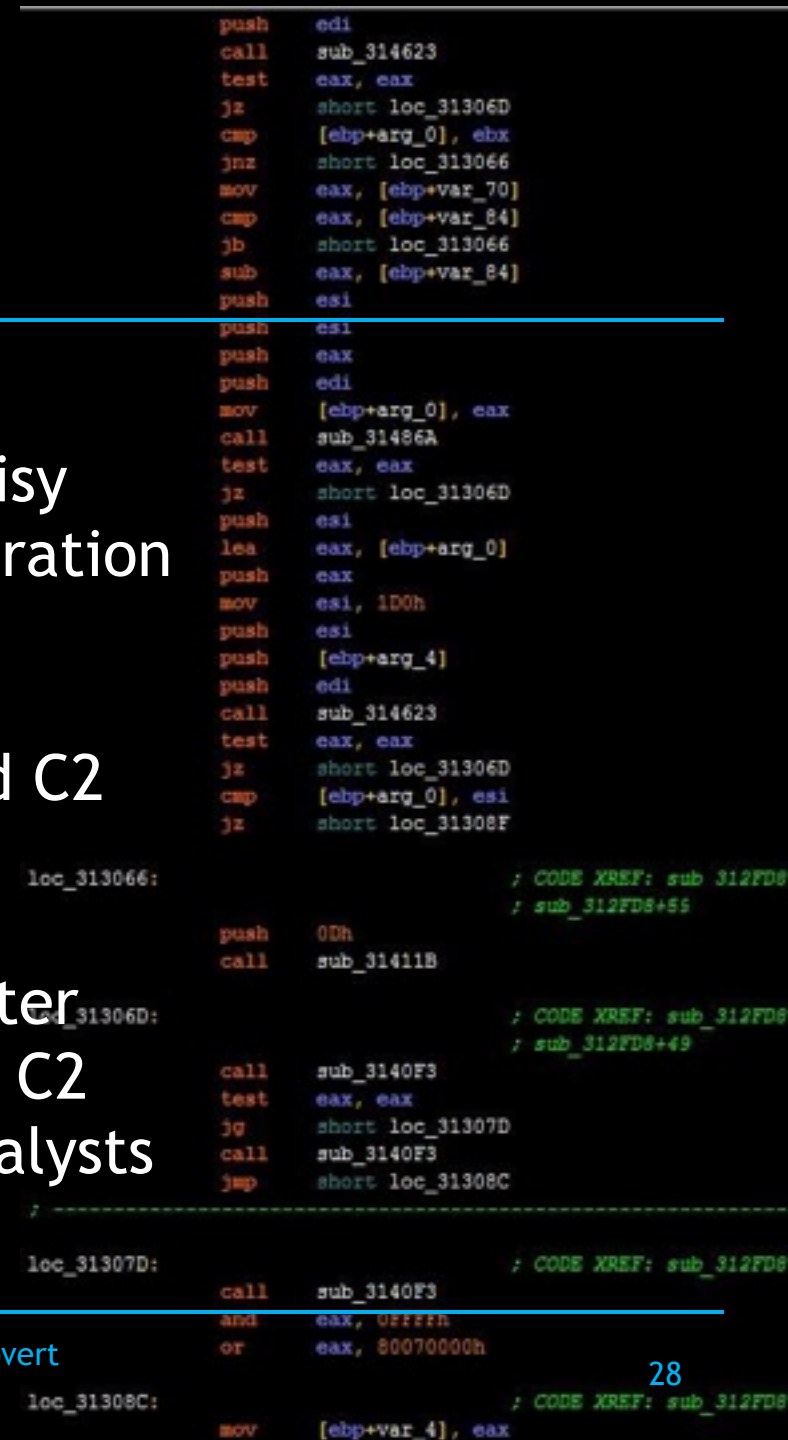
Protocol Tunneling

- Stash our data in the nooks and crannies of an otherwise mundane protocol
  - The goal is to <u>bore the forensic analyst to death</u>
  - Covert data stream must:
    - Make it past the perimeter firewall
    - Blend in with existing traffic



Protocol Tunneling

- HTTP
  - Reliable and flexible, but noisy
  - Typically used for Data exfiltration
- DNS
  - Not suited for data transfer
  - Typically used for WAN-based C2
  - Ubiquitous
- ICMP
  - Often blocked at the perimeter
  - Typically used for LAN-based C2
  - Often Ignored by forensic analysts



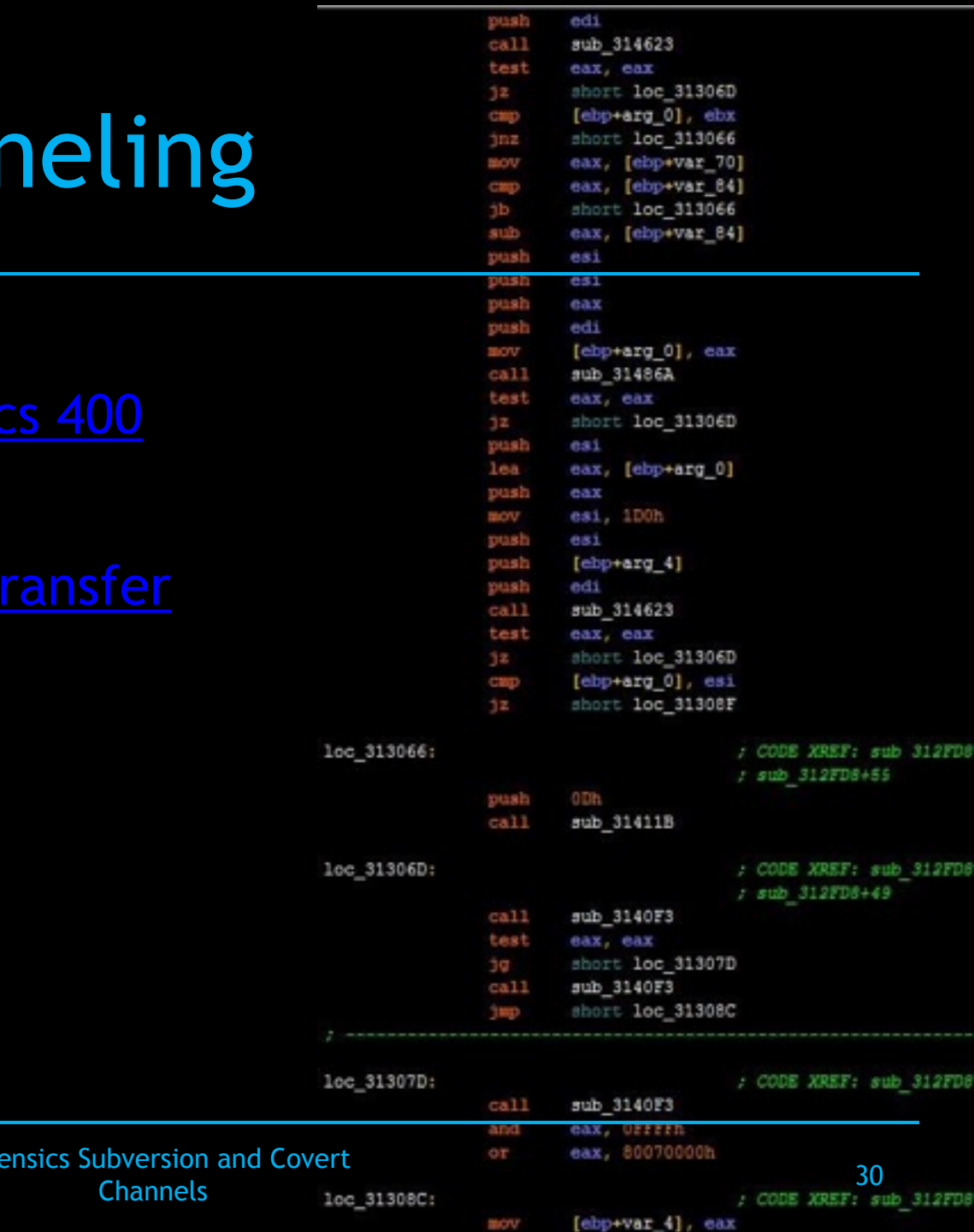
Protocol Tunneling

- ICMP example
  - RC3 CTF Forensics 400

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Protocol Tunneling

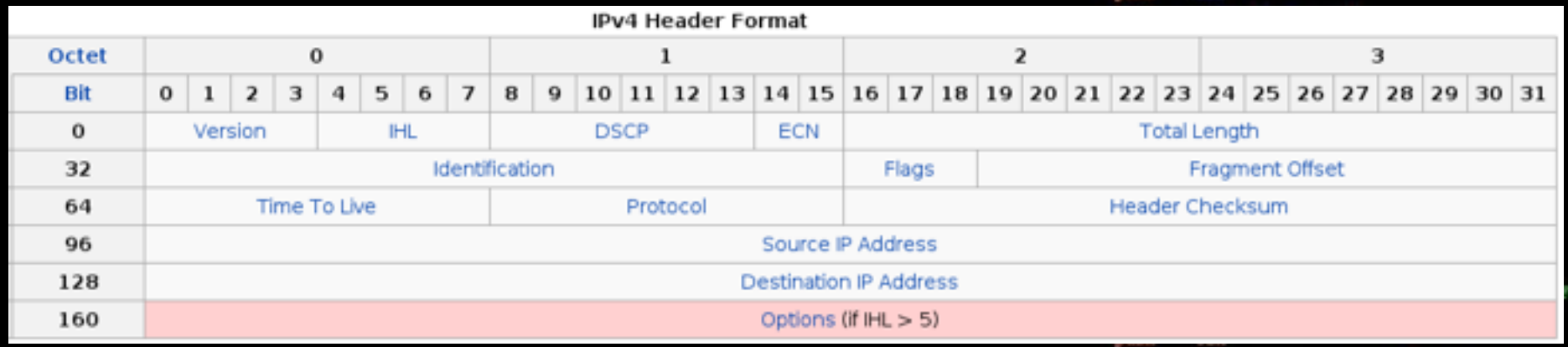
- ICMP example
  - RC3 CTF Forensics 400
- scapy example
  - Kit kappa icmp transfer



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Protocol Tunneling

- IPv4

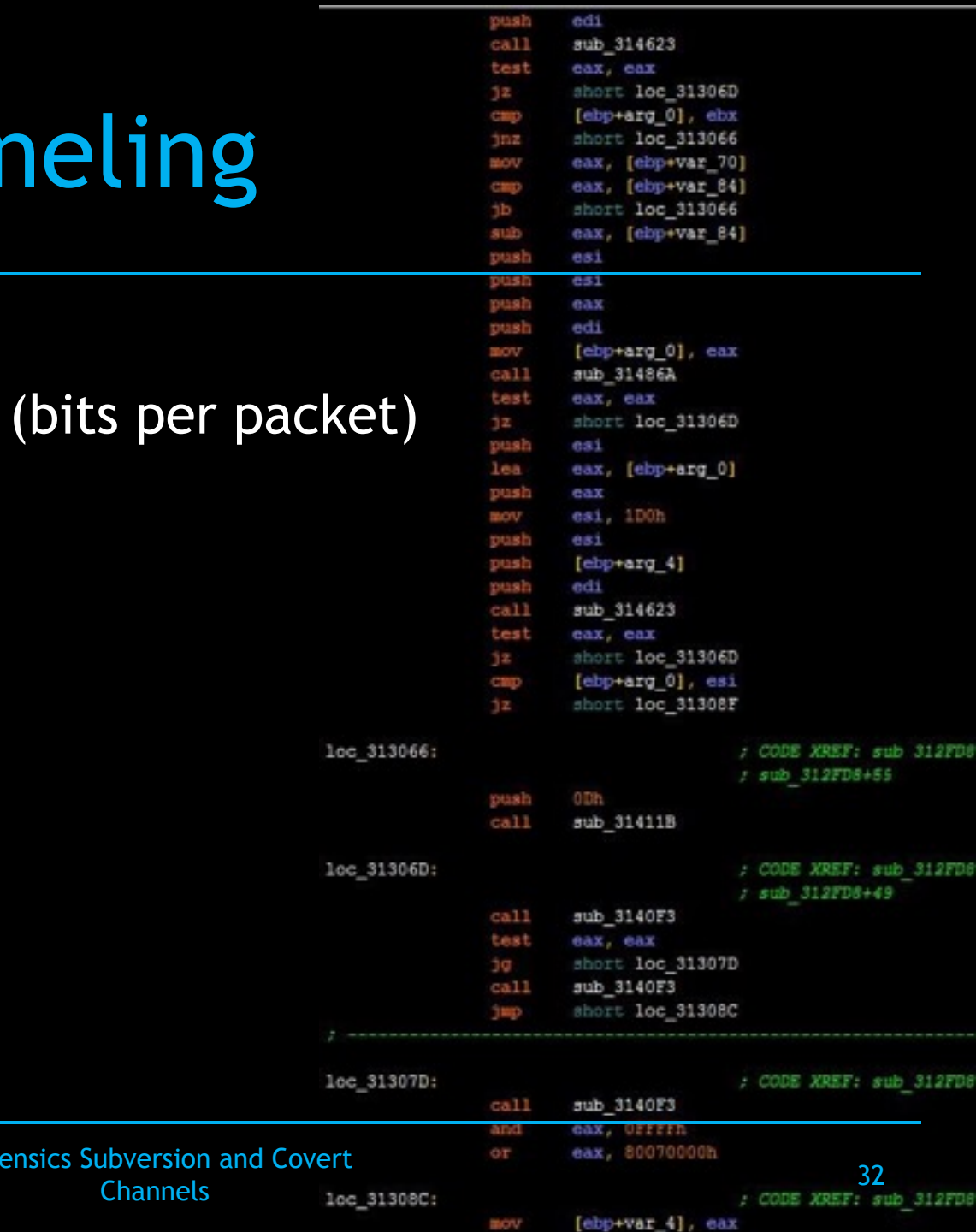


IPv4 Header Format

Octet	0								1								2								3							
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version				IHL				DSCP						ECN		Total Length															
32	Identification																Flags			Fragment Offset											
64	Time To Live								Protocol								Header Checksum															
96	Source IP Address																															
128	Destination IP Address																															
160	Options (if IHL > 5)																															

Protocol Tunneling

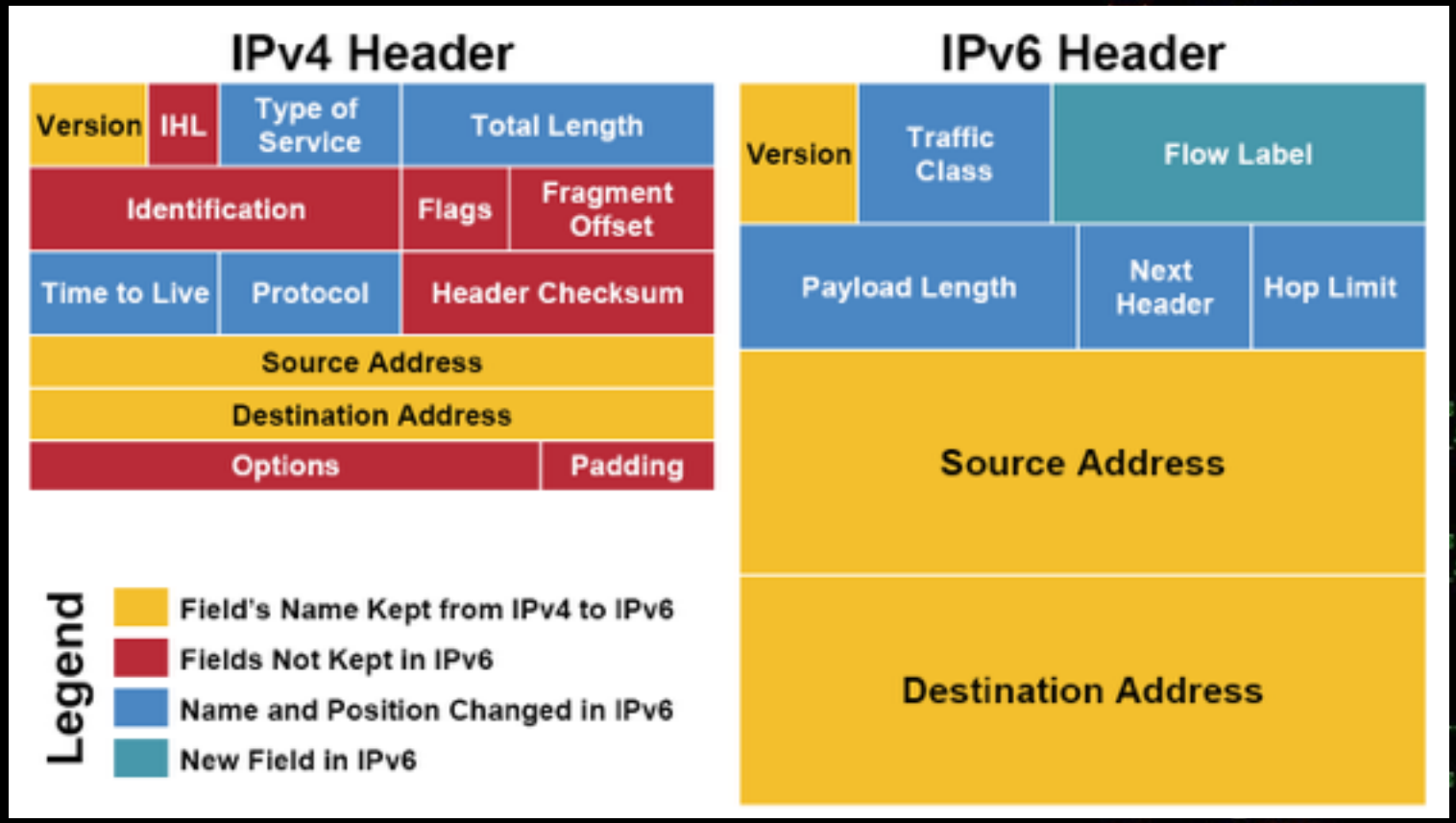
- IPv4
  - ID field - 16 bpp (bits per packet)
  - DF flag - 1 bpp
  - TTL - 8 bpp
  - many more



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Protocol Tunneling

- IPv6



Protocol Tunneling

- IPv6
  - Much simpler...but...

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Protocol Tunneling

- IPv6
  - Much simpler...but...
  - Has extension headers!

Extension Header	Type	Description
Hop-by-Hop Options	0	Options that need to be examined by all devices on the path.
Destination Options (before routing header)	60	Options that need to be examined only by the destination of the packet.
Routing	43	Methods to specify the route for a datagram (used with Mobile IPv6).
Fragment	44	Contains parameters for fragmentation of datagrams.
Authentication Header (AH)	51	Contains information used to verify the authenticity of most parts of the packet.
Encapsulating Security Payload (ESP)	50	Carries encrypted data for secure communication.
Destination Options (before upper-layer header)	60	Options that need to be examined only by the destination of the packet.
Mobility (currently without upper-layer header)	135	Parameters used with Mobile IPv6.

Protocol Tunneling

- IPv6
  - Much simpler...but...
  - Has extension headers!

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
```

***Hop-by-Hop Options* and *Destination Options* extension header format**

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next Header*								*Hdr Ext Len*								*Options and Padding*															
4	32	*Options and Padding*																															
8	64	Optional: more *Options and Padding* ...																															
12	96																																

***Next Header* (8 bits)**

Specifies the type of the next header.

***Hdr Ext Len* (8 bits)**

Length of this header in 8-octet units, not including the first 8 octets.

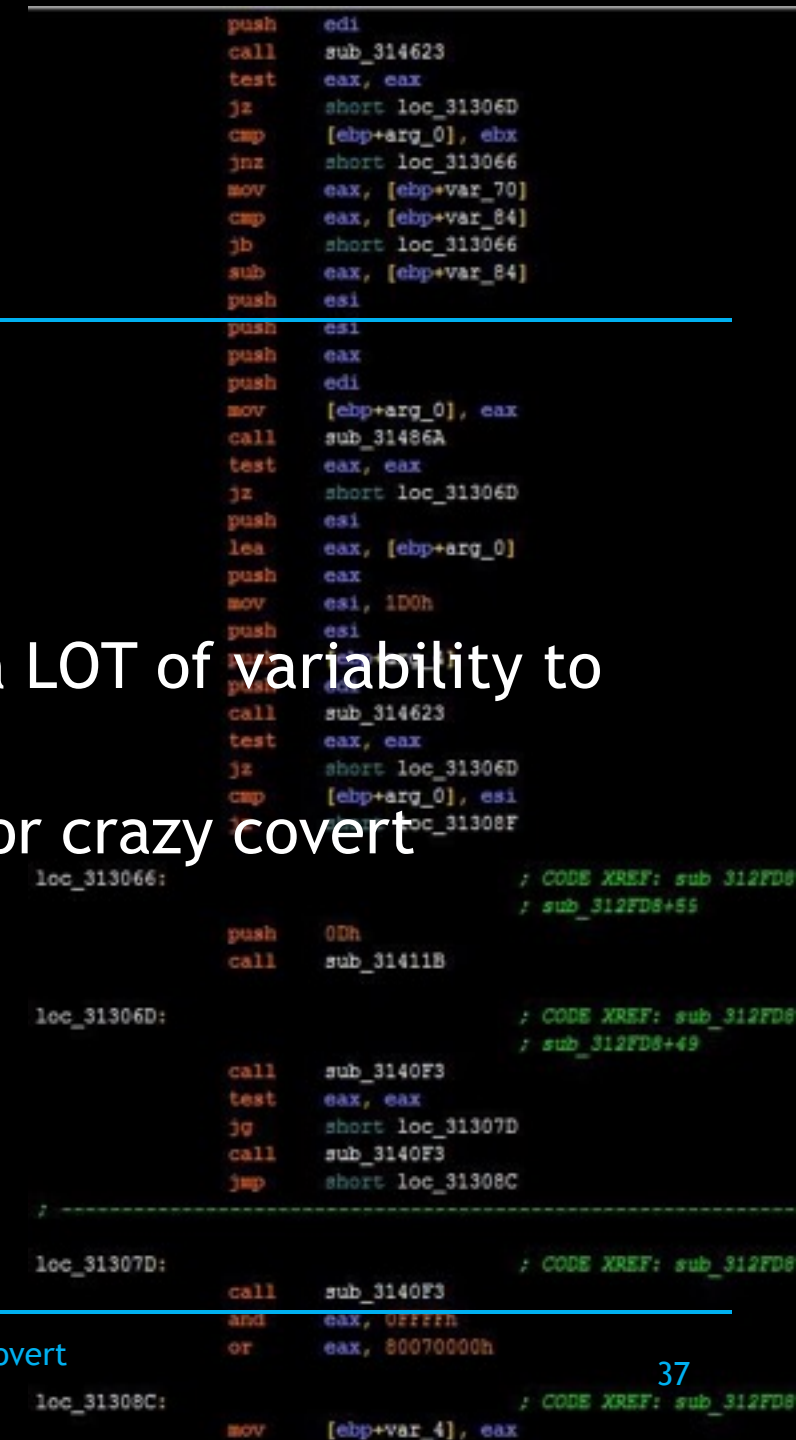
***Options* (variable)**

Contains one or more options, and optional padding fields to align options and to make the total header length a multiple of 8 octets. Options are TLV-coded.

```
jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

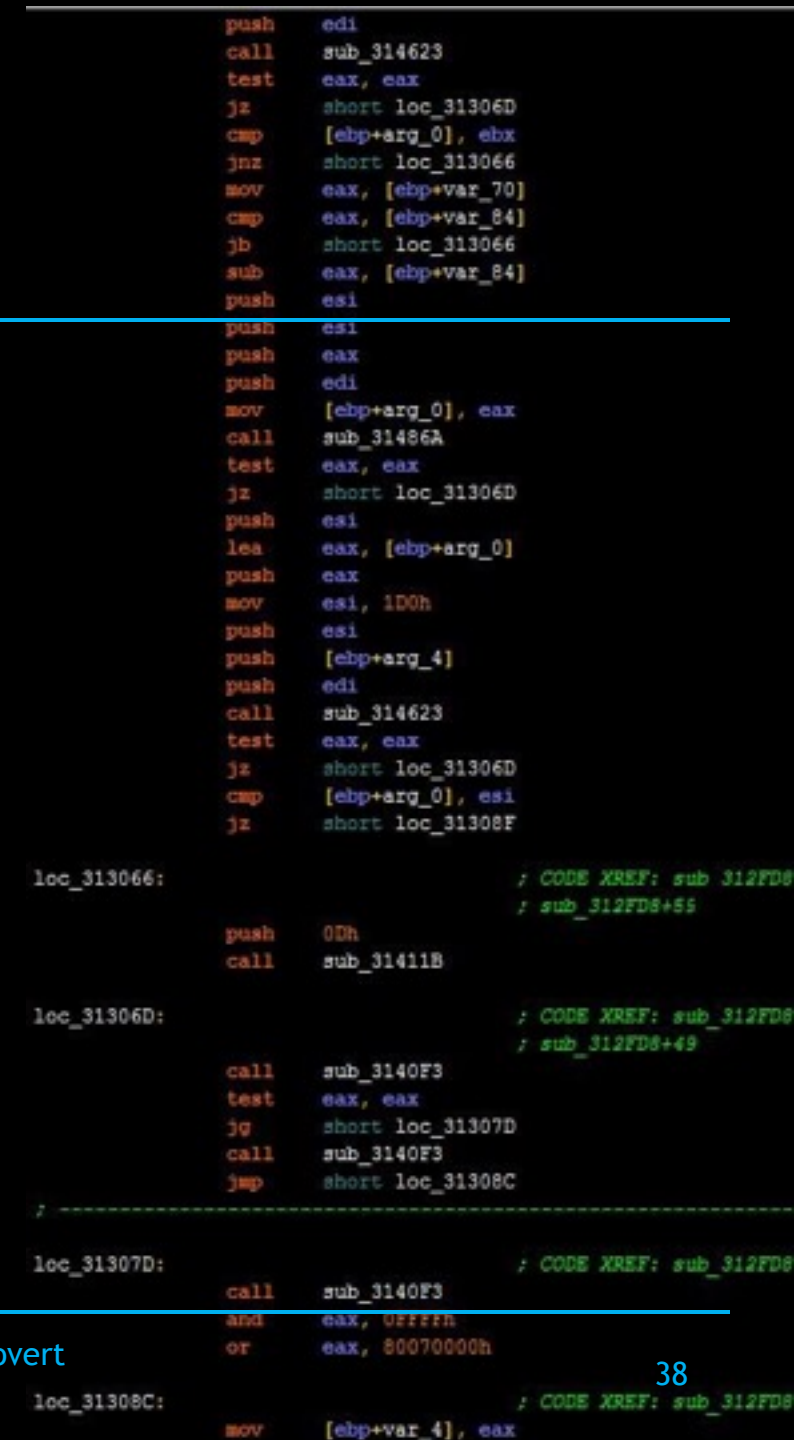
Protocol Tunneling

- IPv6
  - Much simpler...but...
  - Has extension headers!
  - The extension headers add a LOT of variability to each packet
    - Plenty of opportunities for crazy covert channels



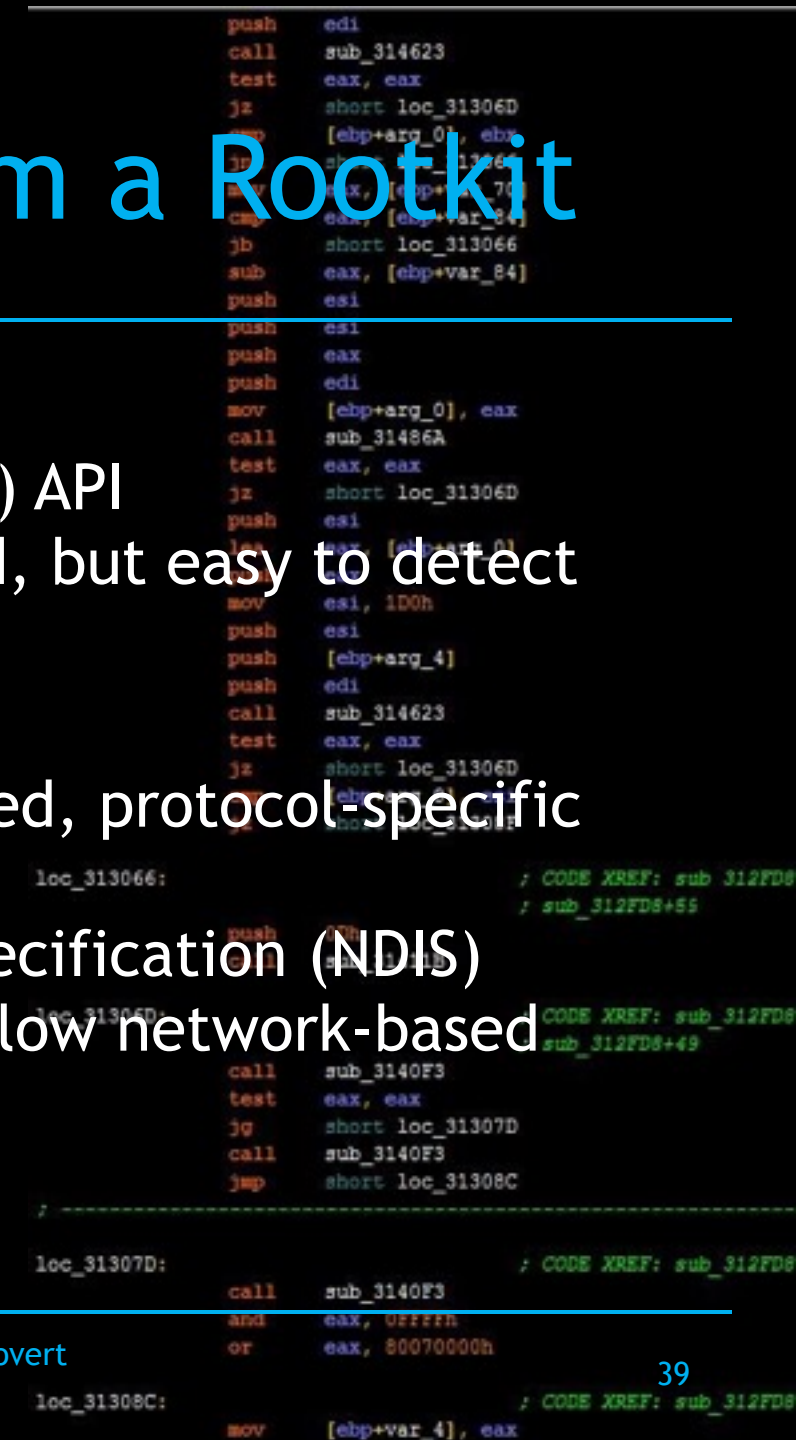
Protocol Tunneling

- Finding covert channels
  - Read the RFCs
  - ...that's about it



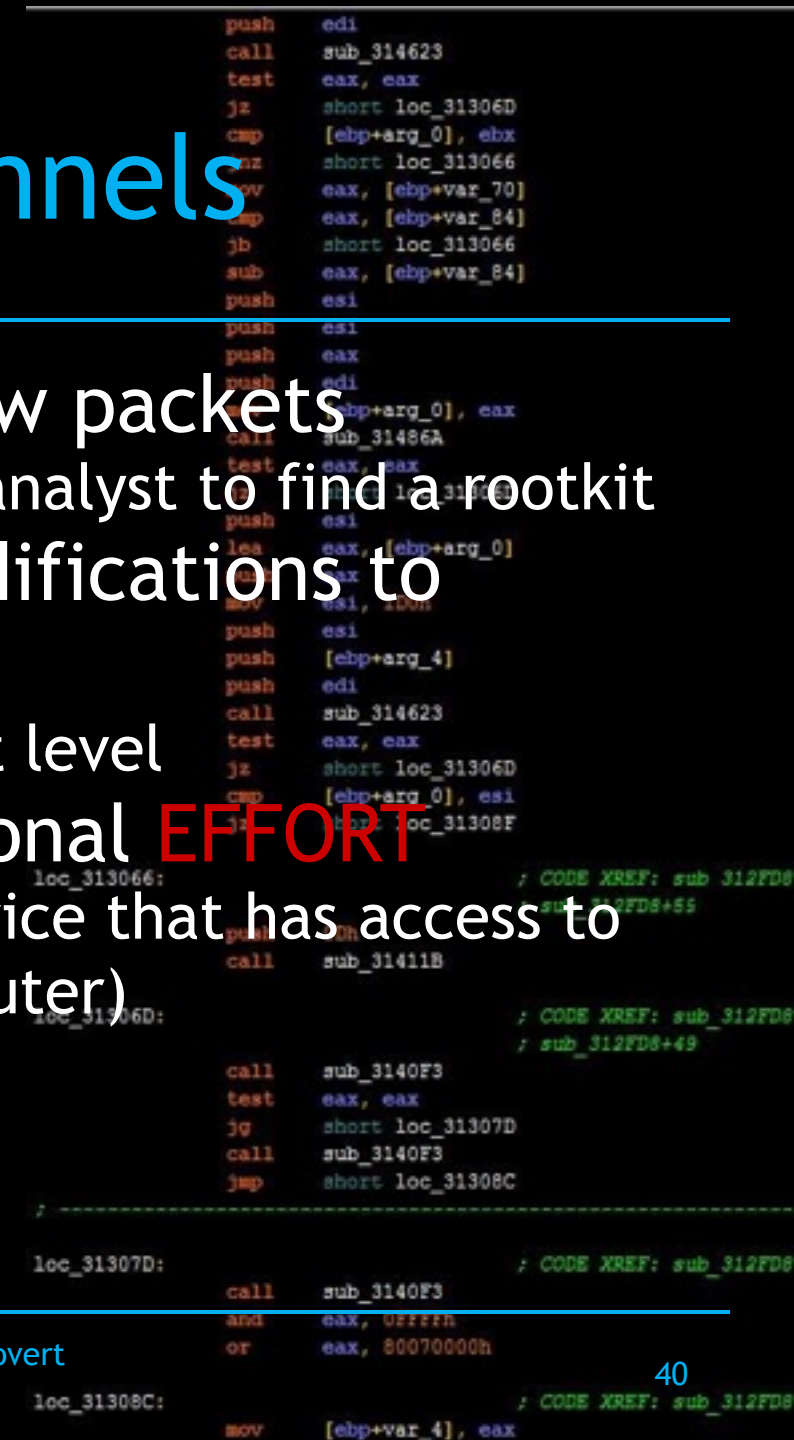
Tunneling Data From a Rootkit

- User-mode
  - Windows Sockets 2 (Winsock) API
    - Simple, well documented, but easy to detect
- Kernel-mode
  - Winsock Kernel (WSK) API
    - More obscure, complicated, protocol-specific details
  - Network Driver Interface Specification (NDIS)
    - High host-based stealth, low network-based stealth



Passive Covert Channels

- You're still generating new packets
  - That may be enough for an analyst to find a rootkit
- Instead, make subtle modifications to existing packets.
  - Steganography at the packet level
- Extra STEALTH but additional EFFORT
  - Requires a foothold on a device that has access to all network traffic (like a router)



Agenda

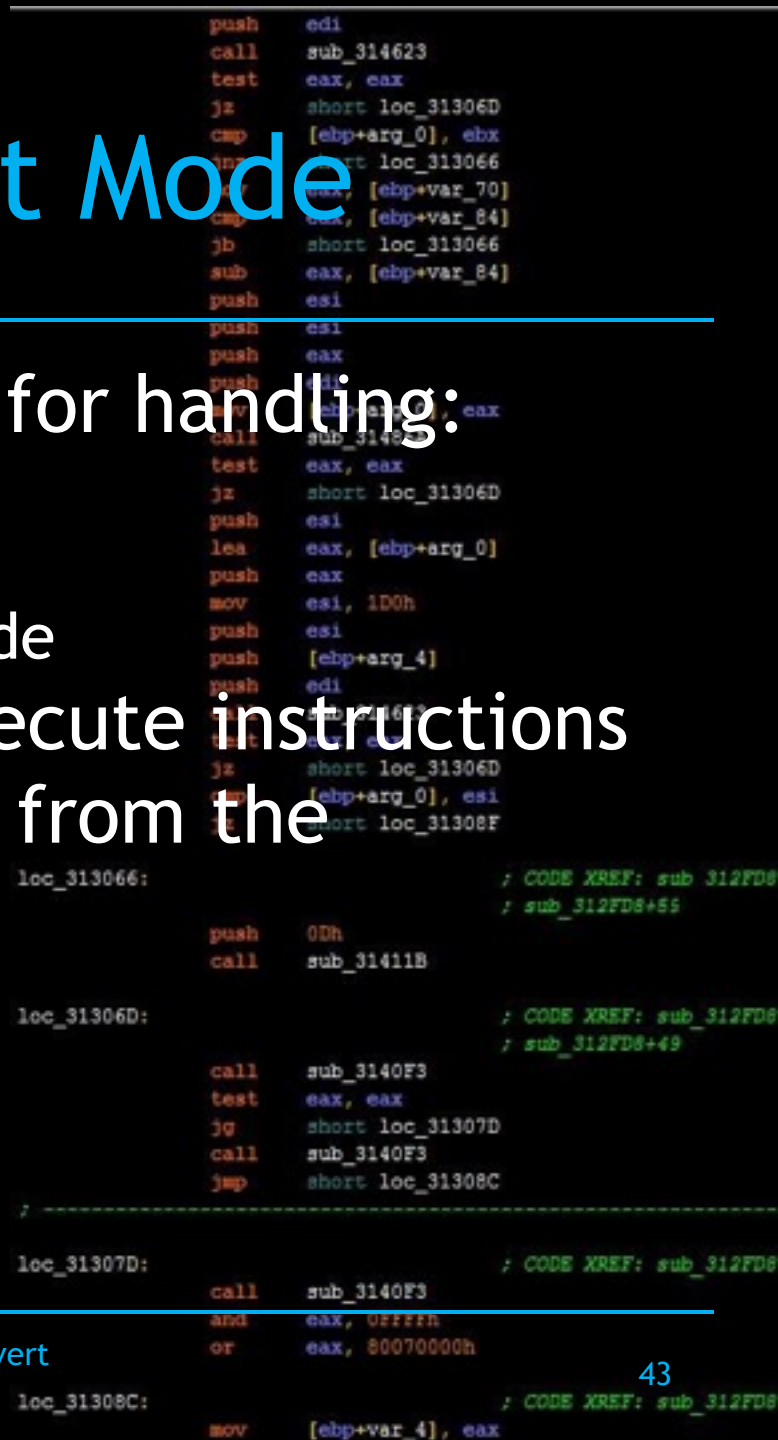
- Access Token Manipulation
- IRP Handling and Filter Drivers
- Forensics Countermeasures
- Covert Channels
- Out-Of-Band

Additional Processor Modes

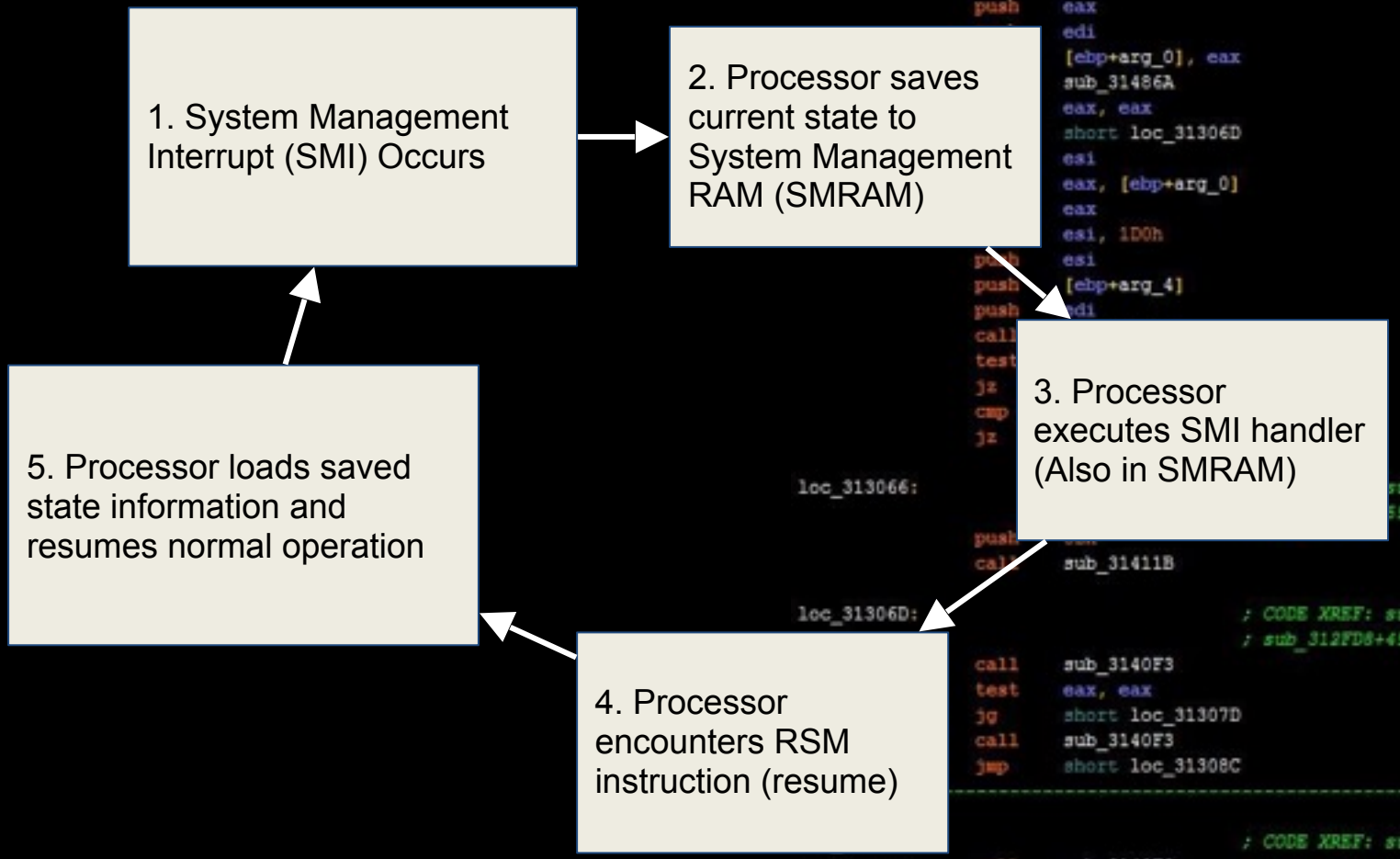
- The IA-32 architecture supports 3 modes:
  - Real mode (kernel mode)
  - Protected mode (user mode)
  - System management mode
- Intel processors that support virtualization have 2 more modes:
  - Root mode
  - Non-root mode

System Management Mode

- Operating mode provided for handling:
  - Power management
  - System hardware control
  - Proprietary OEM designed code
- SMM provides a way to execute instructions in a manner that's hidden from the targeted OS!

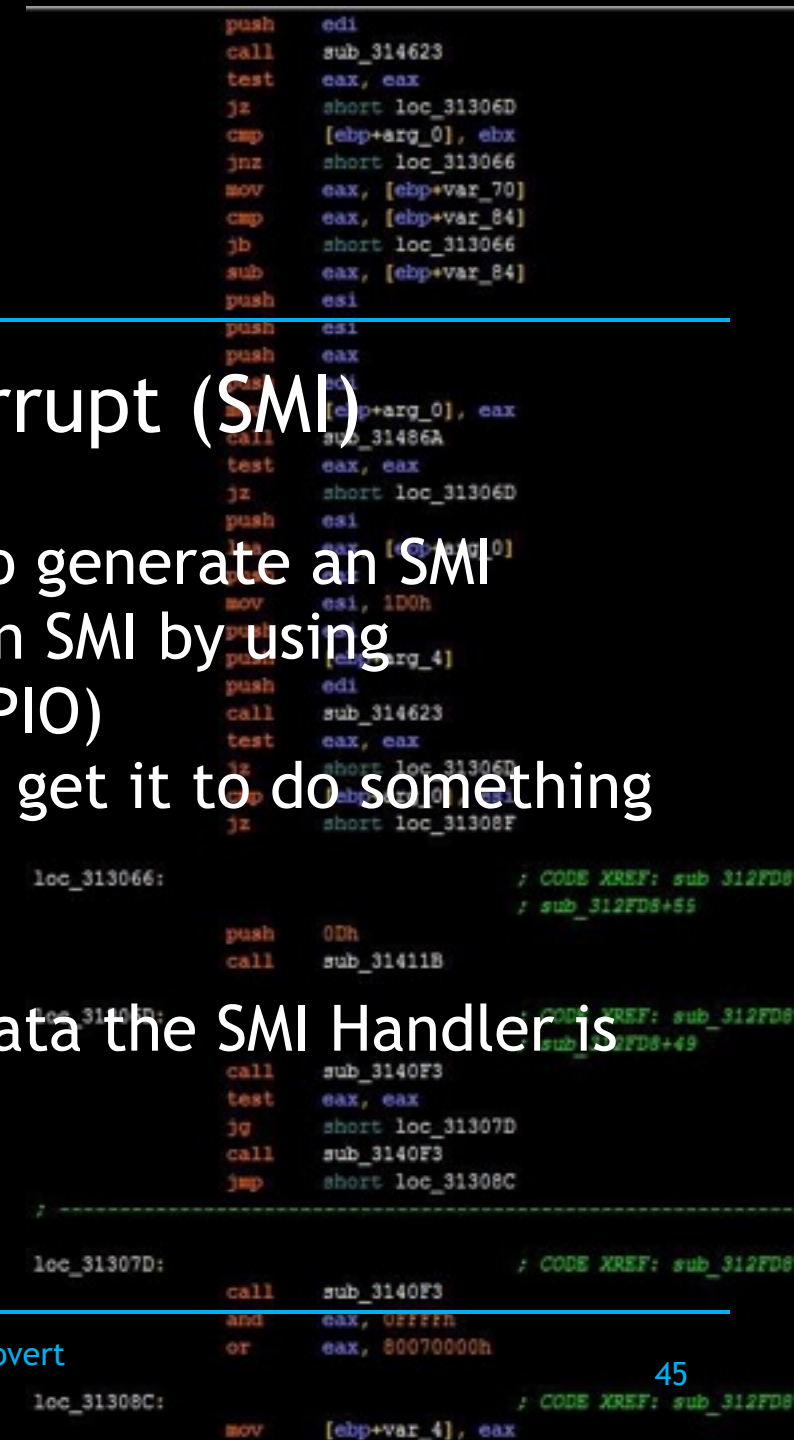


Switching to SMM



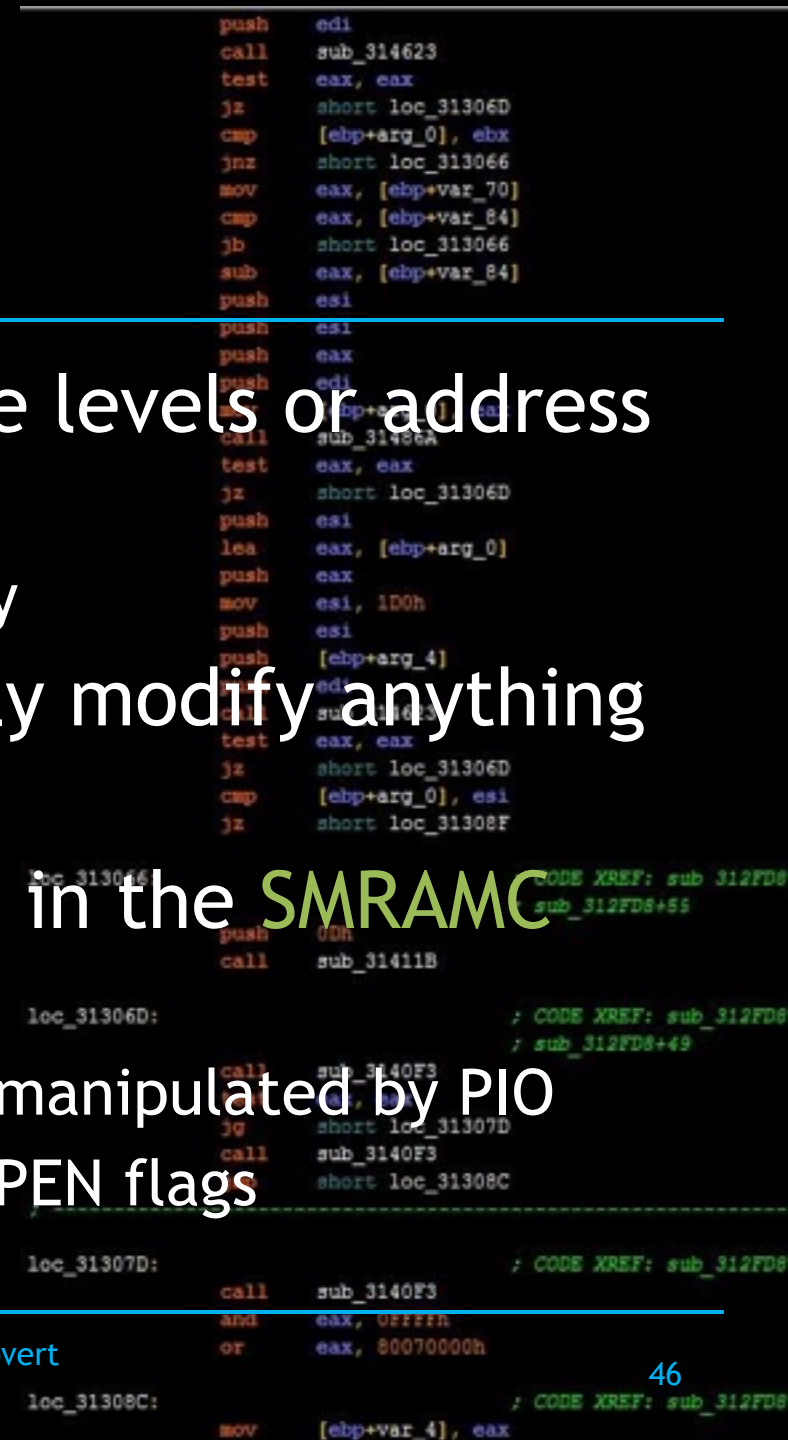
Switching to SMM

- System Management Interrupt (SMI)
  - Normal Hardware Interrupt
  - No hardware-agnostic way to generate an SMI
  - Most researchers generate an SMI by using Programmed Input/Output (PIO)
    - Talk to the hardware and get it to do something to cause an SMI
- SMRAM
  - 64 KB in size, all code and data the SMI Handler is composed of is here



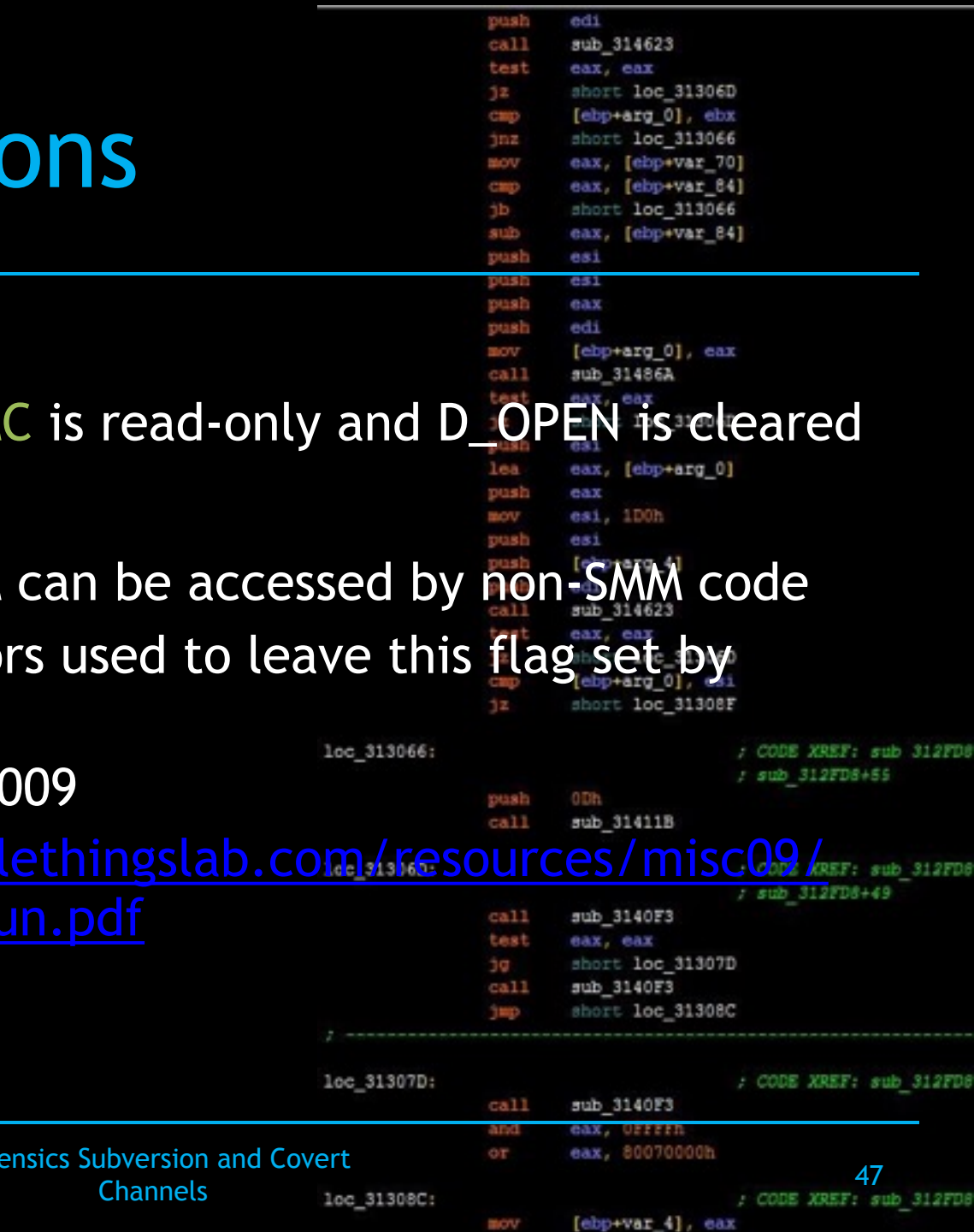
SMM Operations

- No such things as privilege levels or address mappings
  - Just the handler and memory
- The handler can arbitrarily modify anything in physical memory
- SMM is controlled by flags in the SMRAMC Register
  - the SMRAMC register can be manipulated by PIO
  - Contains the D_LCK and D_OPEN flags



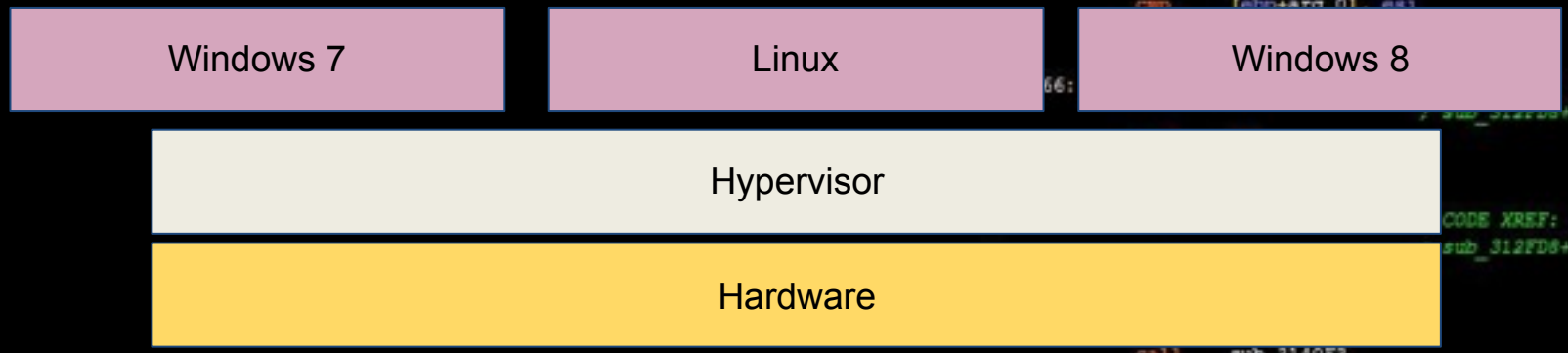
SMM Operations

- D_LCK Flag
  - When set, SMRAMC is read-only and D_OPEN is cleared
- D_OPEN Flag
  - When set, SMRAM can be accessed by non-SMM code
  - Until 2008, vendors used to leave this flag set by accident
    - Bypass found in 2009
      - http://invisiblethingslab.com/resources/misc09/smm_cache_fun.pdf



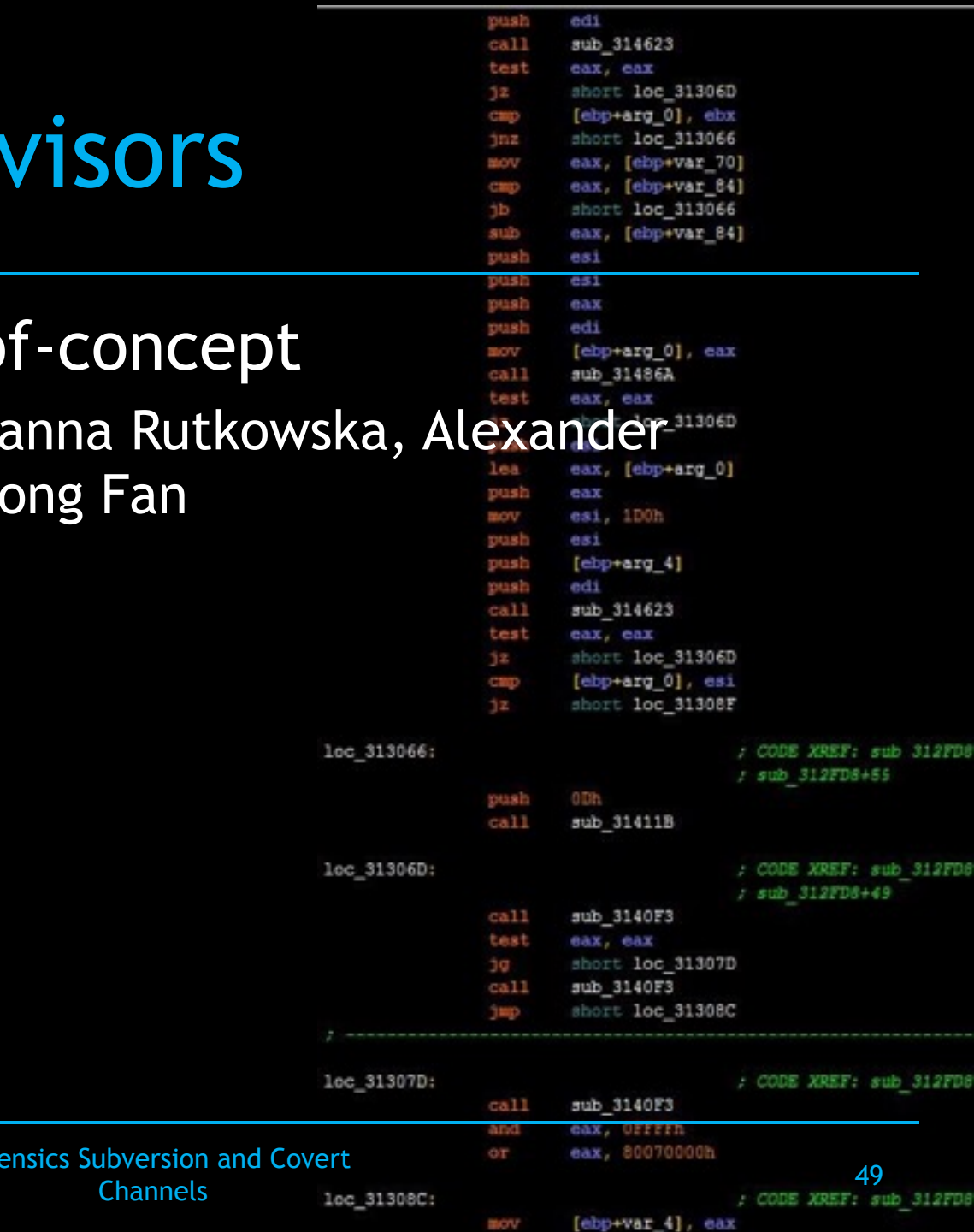
Rogue Hypervisors

- Hypervisor/Virtual Machine Monitor (VMM)
  - Low-level layer of software that allows a computer to share its resources between multiple operating systems



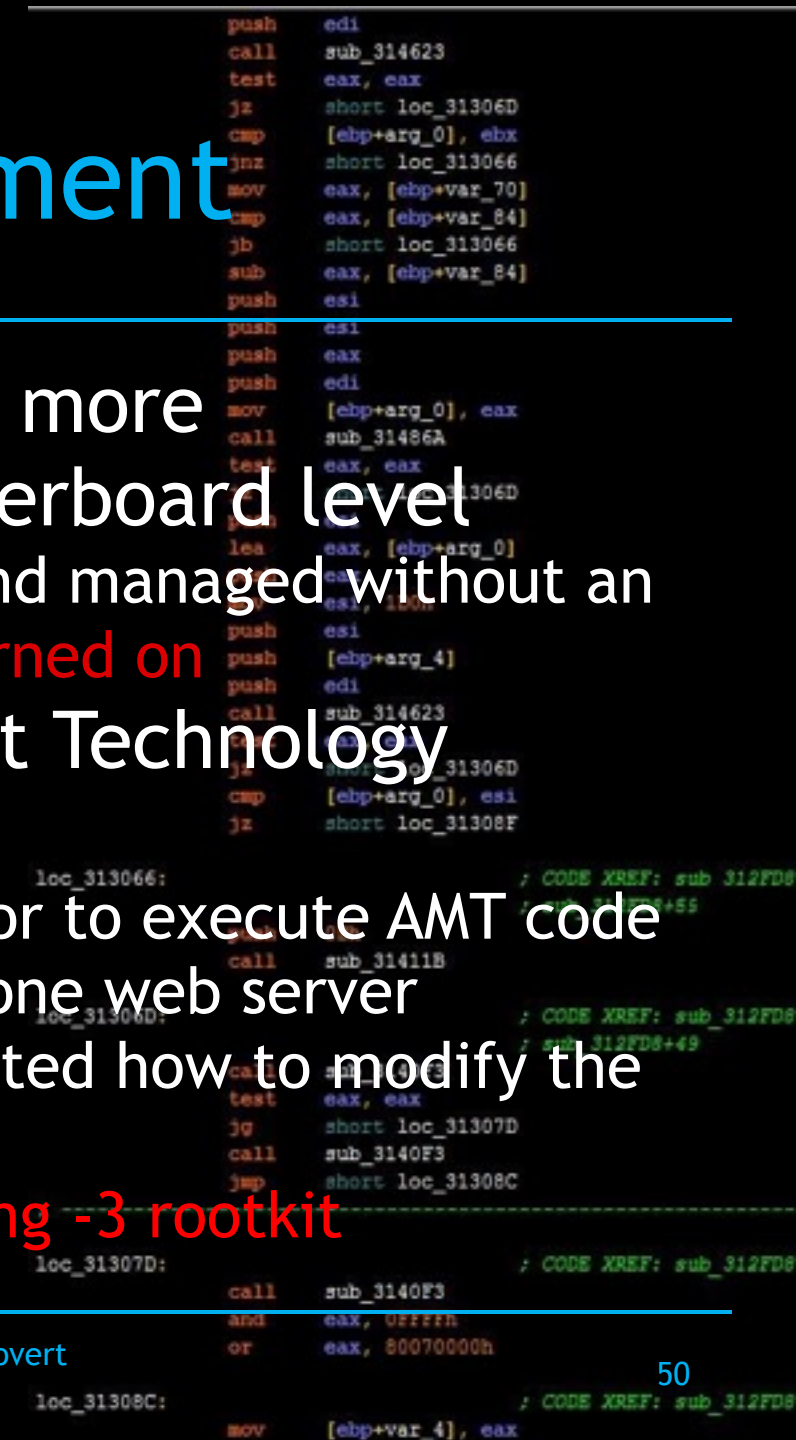
Rogue Hypervisors

- Blue Pill proof-of-concept
  - Developed by Joanna Rutkowska, Alexander Tereshkin, and Rong Fan



Lights-Out Management

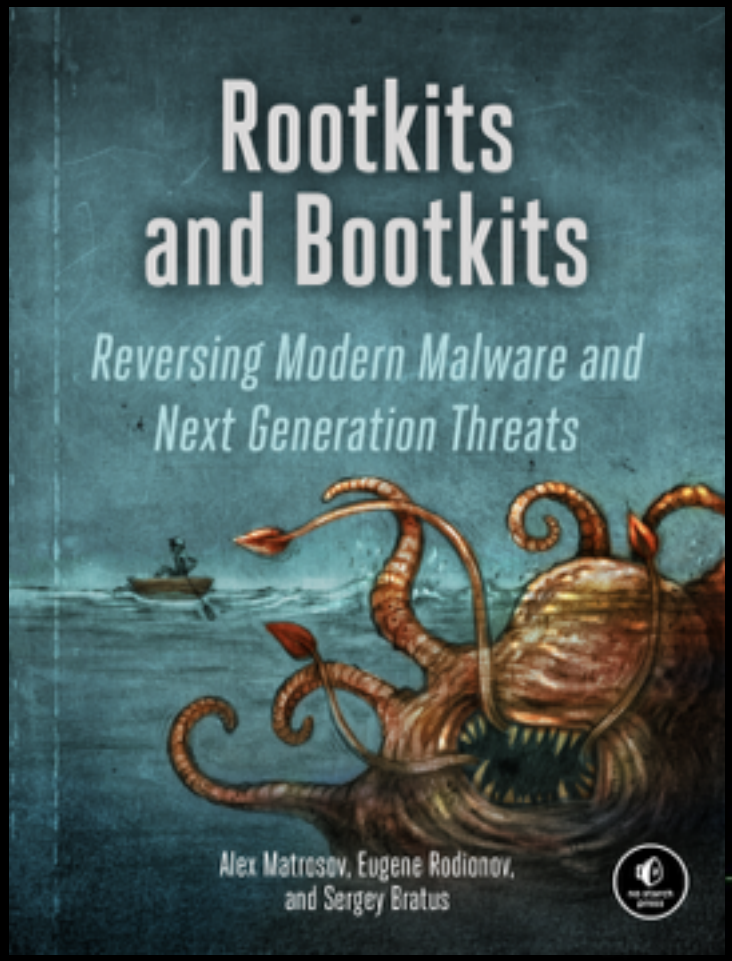
- OEM's have put more and more functionality at the motherboard level
  - Machines can be accessed and managed without an OS or without even being turned on
- Intel's Active Management Technology (AMT)
  - Has an independent processor to execute AMT code
    - Even supports a stand-alone web server
  - Researchers have demonstrated how to modify the code that AMT executes
    - Essentially construct a ring -3 rootkit



Bootkits

- Subvert the boot process so that an attacker's code is executed first
- The next major development in malware
- Mostly used to bypass Microsoft Patchguard
- Unfortunately, outside of the scope of this course

Bootkits




```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

                        ; CODE XREF: sub_312FD8
                        ; sub_312FD8+55
push    0Dh
call    sub_31411B

                        ; CODE XREF: sub_312FD8
                        ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

                        ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:             ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Questions?

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

References

1. Blunden, Bill. The Rootkit Arsenal Escape and Evasion in the Dark Corners of the System, Second Edition. 2nd ed. Burlington, Mass.: Jones & Bartlett Learning, 2013. Print.

1. Hoglund, Greg, and James Butler. Rootkits: Subverting the Windows Kernel. Upper Saddle River, NJ: Addison-Wesley, 2006. Print.