



ハードウェアによる仮想化支援機能を利用したハイパーバイザーIPS

Fourteenforty Research Institute, Inc.

株式会社 フォティーンフォティ技術研究所

<http://www.fourteenforty.jp>



はじめに(1/2)

- 近年、ルートキット技術の利用によりマルウェアのステルス化が進んでいる
- 従来のAVプロダクトによる検出が困難
- 3rd Partyによる「検出ツール」
 - GMER
 - Rootkit Revealer(Microsoft)
 - Rootkit Buster(TrendMicro)
 - McAfee Rootkit Detective



はじめに(1/2)

- 近年、ルートキット技術の利用によりマルウェアのステルス化が進んでいる
- 従来のAVプロダクトによる検出が困難
- 3rd Partyによる「検出ツール」
 - GMER
 - Rootkit Revealer(Microsoft)
 - Rootkit Buster(TrendMicro)
 - McAfee Rootkit Detective
 - Windbg ☺



はじめに(2/2)

- ・ 仮想化技術を利用したIPSを開発
 - OSより低い「リング-1」相当の権限で動作
- ・ ルートキットの「検知」と「防止」を実現
 - 不正なメモリパッチングを検知・防止
 - (ゲスト)OSの活動をモニタリング



アウトライン

- ・ 既存のルートキット技術
- ・ 仮想化技術の概要
- ・ Viton, ハイパーバイザーIPS

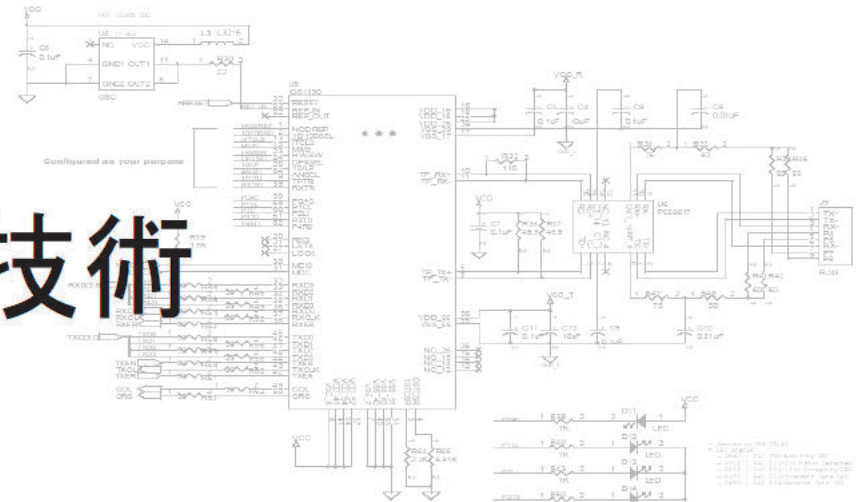


```
int packet_analysis(GDDCONFIG *gdd, unsigned char *packet, unsigned long length)
```

```
struct ip      *ip_header; /* IP header */
struct tcphdr *tcp_header; /* TCP header */
struct tcpopt *tcp_data; /* TCP data */
struct in_addr src_addr; /* Source IP address */
struct in_addr dst_addr; /* Destination IP address */
unsigned short sourcePort; /* Source Port */
unsigned short destPort; /* Destination Port */
unsigned long len_data; /* Length of data part */
unsigned long iph_len; /* Length of IP header */
unsigned long tcph_len; /* Length of TCP header */
...
static char datestr[512]; /* Buffer to store datetime */
time_t timeval;
struct tm *timep=NULL;
char *timesp=NULL;
char *tc;
...
/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_p!=IPPROTO_TCP)
[ ] ip_header->ip_v!=4) return(0);
iph_len = ((unsigned long) (ip_header->ip_hl))+4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) [
return(0);
]
/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcph_len = ((unsigned long) (tcp_header->th_off))+4;
tcp_data = (char *) tcp_header+tcph_len;
if (tcph_len<MINSIZE_TCP) return(0);
/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcph_len)>0)
return(0);
len_data = (unsigned long)ntohs(ip_header->ip_len)
-iph_len-tcph_len;
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_src), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
strcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
```

Chapter 1

既存のルータキット技術



00001870	FF 15 F0 11 00 01 E9 00 03 00 00 E8 78 10 00 00	3. 離
00001880	33 F8 56 E8 B1 F0 FF 85 00 0F 84 B7 03 00 00	Vj. 51
00001890	56 6A 02 FF 35 60 80 00 01 FF 35 D0 87 00 01 FF	2. ... u.h. 5
000018A0	15 0C 11 00 01 85 00 75 10 68 10 10 00 00 FF 35	P... 50 52
000018B0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	52
000018C0	FF 15 04 12 00 01 FF 35 00 87 00 01 FF 15 2C 12	5p. ... 誠
000018D0	00 01 FF 35 04 88 00 01 FF 15 68 10 00 01 E9 64	山
000018E0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	3. 95
000018F0	11 0F 85 16 01 00 00 03 F6 39 35 E8 87 00 01 74	5P. 5. 意
00001900	22 8B 30 28 12 00 01 56 FF D7 56 FF D7 68 00 10	5. ... 意
00001910	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 70	5P. 5. 意
00001920	02 00 00 6A 01 E8 0F FG FF FF E9 1A 00 00 8B	5P. 5. 意
00001930	70 14 88 11 01 00 00 25 F0 0F 87 85 00 00 25	5P. 5. 意
00001940	F0 0F 84 16 02 00 00 83 FE 1C 0F 85 8D 00 00 00	5P. 5. 意
00001950	33 F8 39 75 10 74 2F A1 EC 87 00 01 8B 00 F0 87	5P. 5. 意
00001960	00 01 38 06 75 08 38 CE 0F 84 D9 02 00 00 8B 3D	5P. 5. 意
00001970	14 12 00 01 51 50 68 B1 00 00 00 FF 35 04 87 00	5P. 5. 意
00001980	01 E9 56 01 00 00 2B 2D 14 12 00 01 68 F0 87 00	5P. 5. 意
00001990	01 68 EC 87 00 01 68 80 00 00 FF 35 04 87 00	5P. 5. 意
000019A0	01 FF D7 A1 EC 87 00 01 8B 00 F0 87 00 01 3B C1	5P. 5. 意
000019B0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	5P. 5. 意
000019C0	02 00 00 51 50 E9 07 01 00 00 8B CE 88 12 01 00	5P. 5. 意
000019D0	00 2B 00 0F 84 30 02 00 00 83 E9 04 0F 84 29 02	5P. 5. 意
000019E0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 0F	5P. 5. 意
000019F0	84 E9 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	5P. 5. 意
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 60	5P. 5. 意
00001D10	88 09 01 0F 85 EE 00 00 00 25 45 14 8B 48 0C 8B	5P. 5. 意
00001D20	01 8B 01 D7 80 01 EA 02 83 E0 01 83 E2 01 F6 01	5P. 5. 意



ルートキット技術の分類

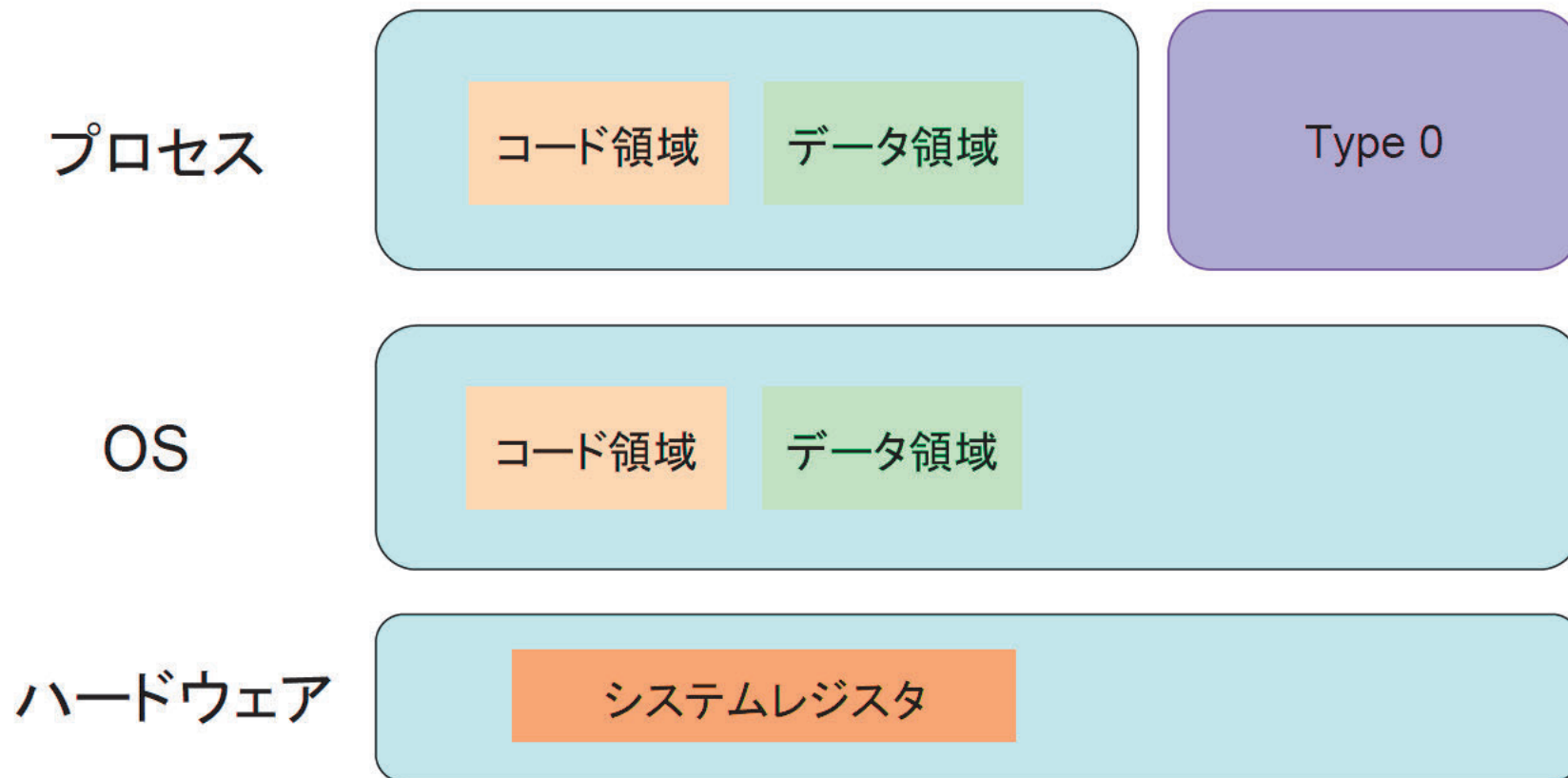
ステルスマルウェア

ユーザーの同意なしに、システム・OSの挙動を変更し、情報の隠ぺいを行うソフトウェア(マルウェア)

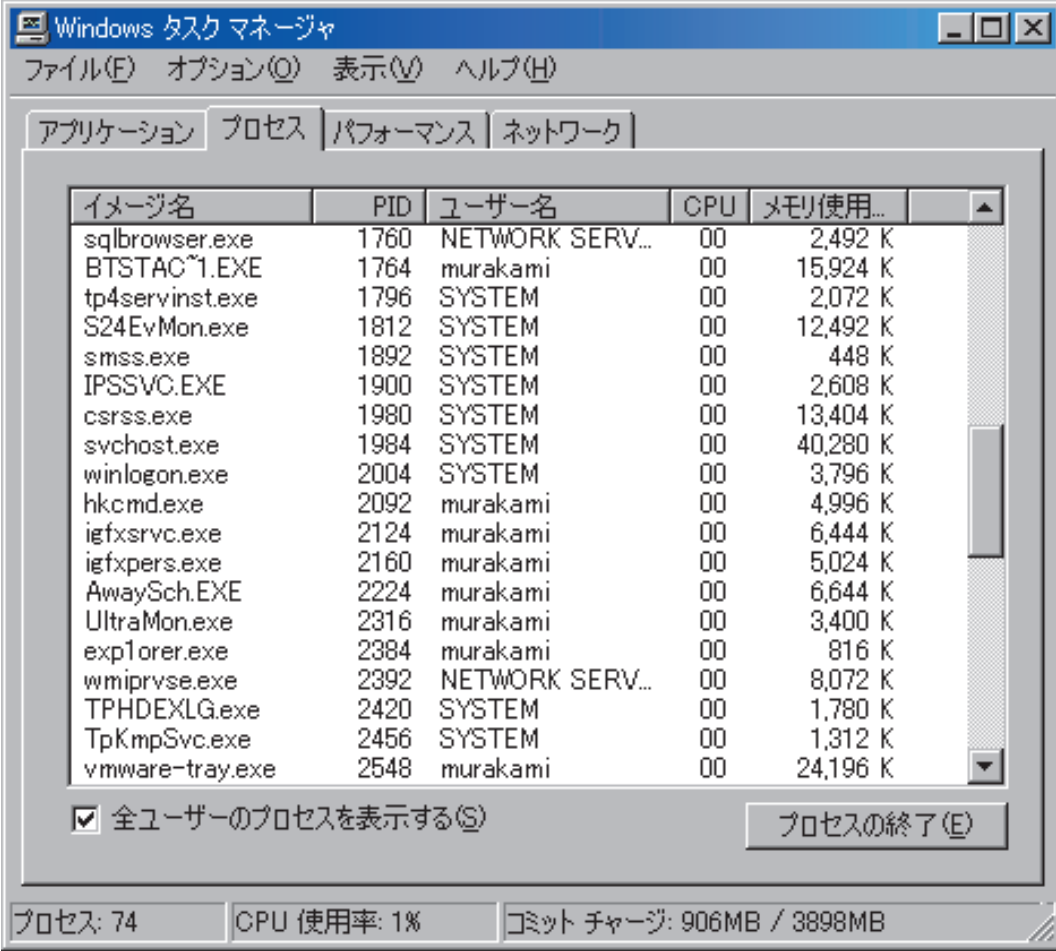
※Joanna Rukowska, "Introducing Stealth Malware Taxnomy"
<http://invisiblethings.org/papers/malware-taxonomy.pdf>

ステルスマルウェアがパッチングを行うリソースの性質に基づいて
Type 0からType IIIの4種類に分類

Type 0 (≠ステルスマルウェア)



Type 0 (≠ステルスマルウェア)



Windows タスク マネージャ

ファイル(F) オプション(O) 表示(V) ヘルプ(H)

アプリケーション プロセス パフォーマンス ネットワーク

イメージ名	PID	ユーザー名	CPU	メモリ使用...
sqlbrowser.exe	1760	NETWORK SERV...	00	2,492 K
BTSTAC~1.EXE	1764	murakami	00	15,924 K
tp4servinst.exe	1796	SYSTEM	00	2,072 K
S24EvMon.exe	1812	SYSTEM	00	12,492 K
smss.exe	1892	SYSTEM	00	448 K
IPSSVC.EXE	1900	SYSTEM	00	2,608 K
csrss.exe	1980	SYSTEM	00	13,404 K
svchost.exe	1984	SYSTEM	00	40,280 K
winlogon.exe	2004	SYSTEM	00	3,796 K
hkcmd.exe	2092	murakami	00	4,996 K
igfxsrv.exe	2124	murakami	00	6,444 K
igfxpers.exe	2160	murakami	00	5,024 K
AwaySch.EXE	2224	murakami	00	6,644 K
UltraMon.exe	2316	murakami	00	3,400 K
explorer.exe	2384	murakami	00	816 K
wmiprvse.exe	2392	NETWORK SERV...	00	8,072 K
TPHDEXLG.exe	2420	SYSTEM	00	1,780 K
TpKmpSvc.exe	2456	SYSTEM	00	1,312 K
vmware-tray.exe	2548	murakami	00	24,196 K

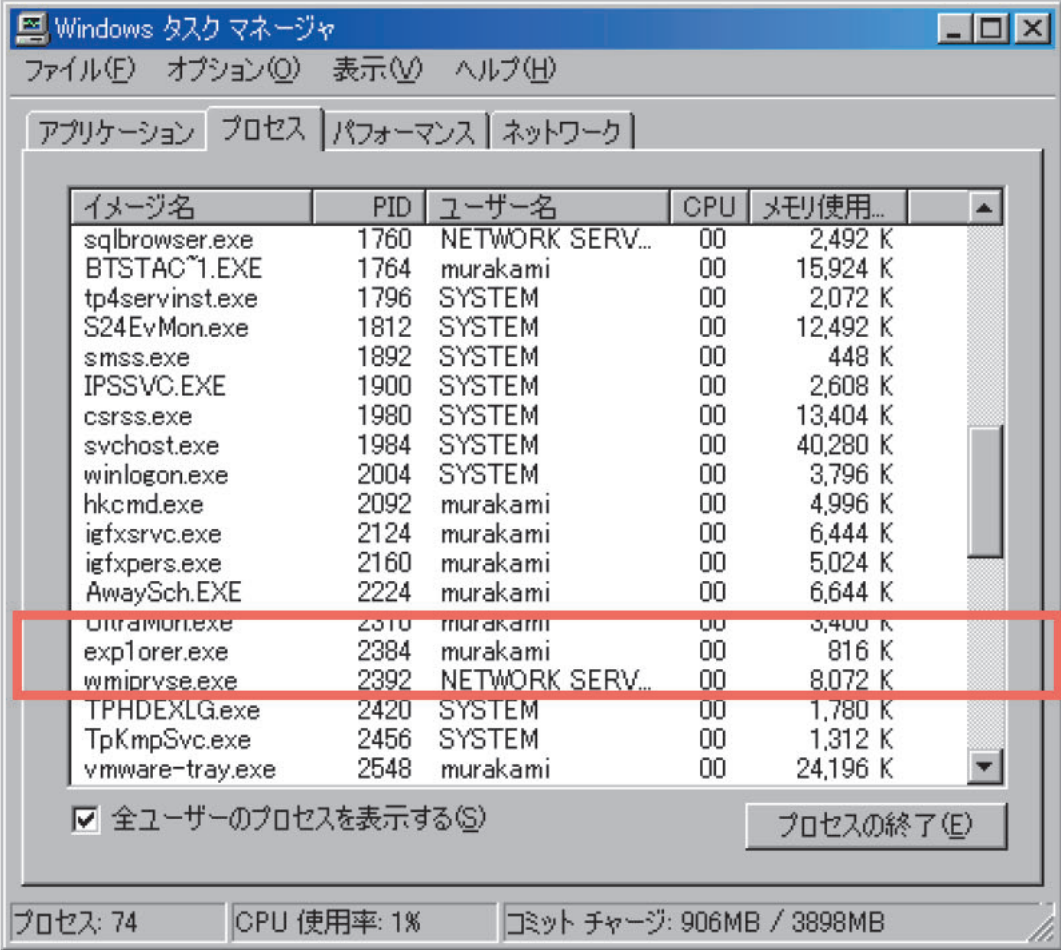
全ユーザーのプロセスを表示する(S)

プロセスの終了(E)

プロセス: 74 CPU 使用率: 1% コミット チャージ: 906MB / 3898MB

Type 0 (≠ステルスマルウェア)

explorer.exe



Windows タスク マネージャ

ファイル(F) オプション(O) 表示(V) ヘルプ(H)

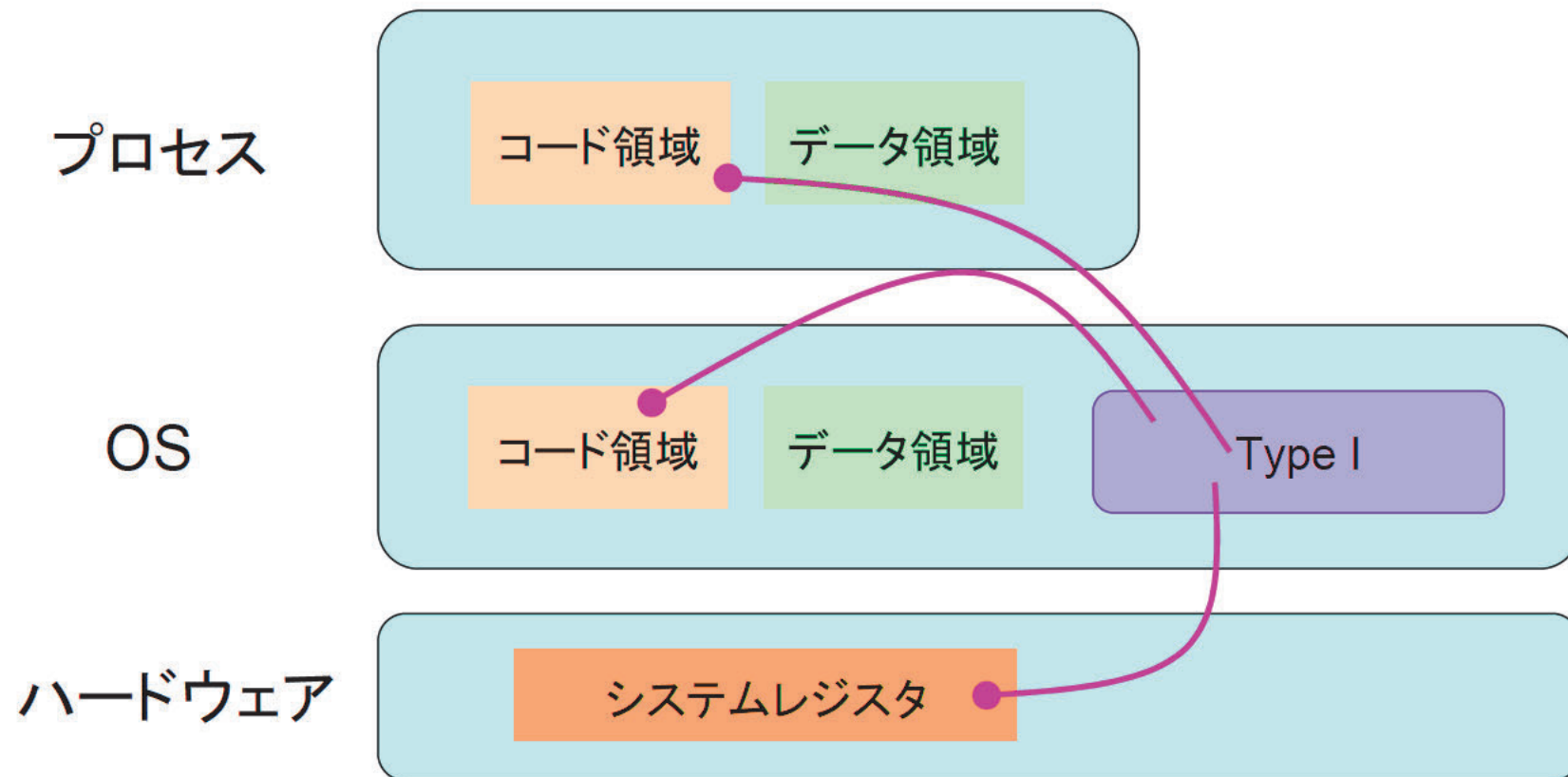
アプリケーション プロセス パフォーマンス ネットワーク

イメージ名	PID	ユーザー名	CPU	メモリ使用...
sqlbrowser.exe	1760	NETWORK SERV...	00	2,492 K
BTSTAC~1.EXE	1764	murakami	00	15,924 K
tp4servinst.exe	1796	SYSTEM	00	2,072 K
S24EvMon.exe	1812	SYSTEM	00	12,492 K
smss.exe	1892	SYSTEM	00	448 K
IPSSVC.EXE	1900	SYSTEM	00	2,608 K
csrss.exe	1980	SYSTEM	00	13,404 K
svchost.exe	1984	SYSTEM	00	40,280 K
winlogon.exe	2004	SYSTEM	00	3,796 K
hkcmd.exe	2092	murakami	00	4,996 K
igfxsrv.exe	2124	murakami	00	6,444 K
igfxpers.exe	2160	murakami	00	5,024 K
AwaySch.EXE	2224	murakami	00	6,644 K
Unramon.exe	2310	murakami	00	3,400 K
explorer.exe	2384	murakami	00	816 K
wmiprivse.exe	2392	NETWORK SERV...	00	8,072 K
TPHDEXLG.exe	2420	SYSTEM	00	1,780 K
TpKmpSvc.exe	2456	SYSTEM	00	1,312 K
vmware-tray.exe	2548	murakami	00	24,196 K

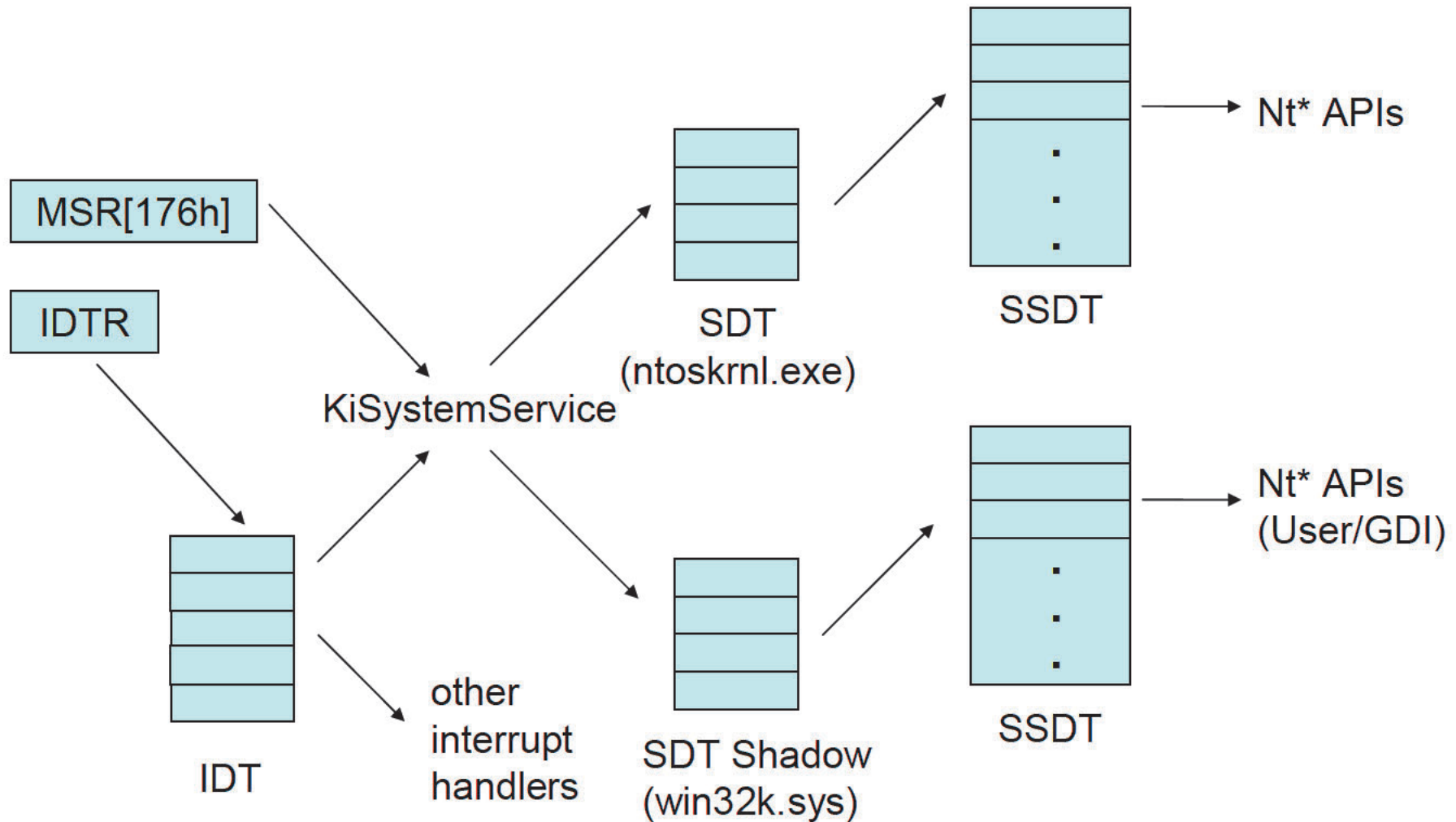
全ユーザーのプロセスを表示する(S) プロセスの終了(E)

プロセス: 74 CPU 使用率: 1% コミット チャージ: 906MB / 3898MB

Type I

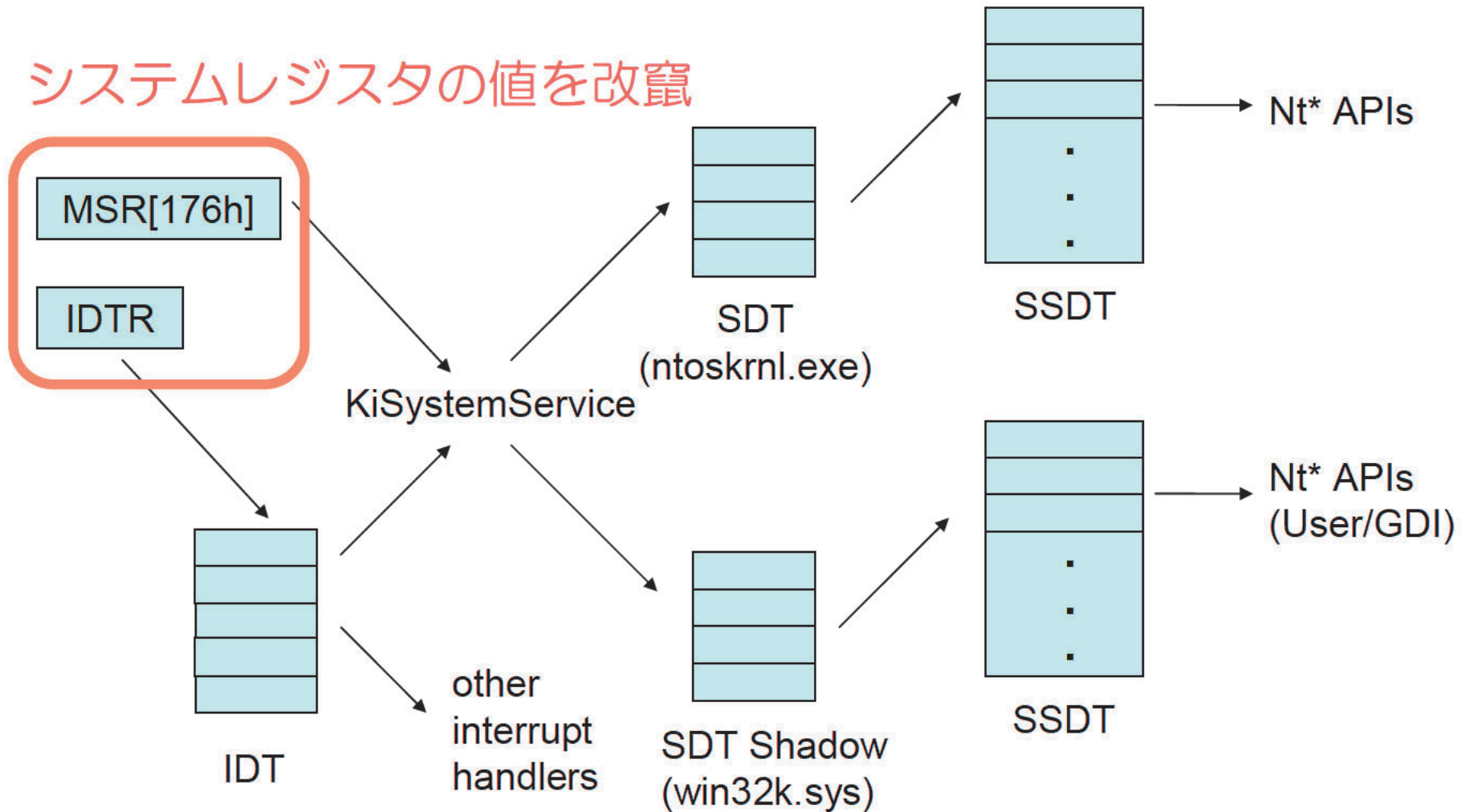


Type I – Windowsにおけるフッキングポイント(1/3)



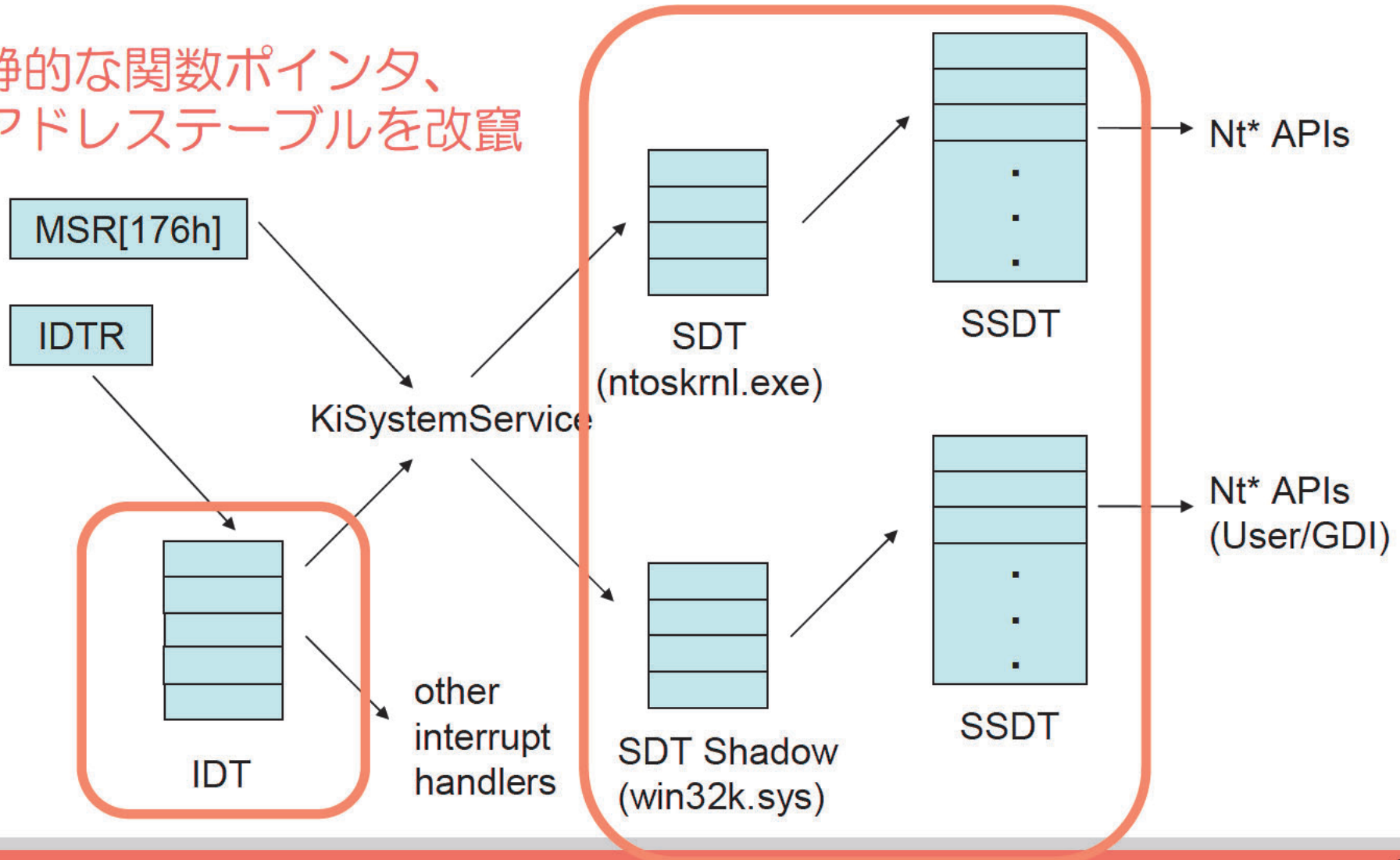
Type I – Windowsにおけるフッキングポイント(1/3)

システムレジスタの値を改竄



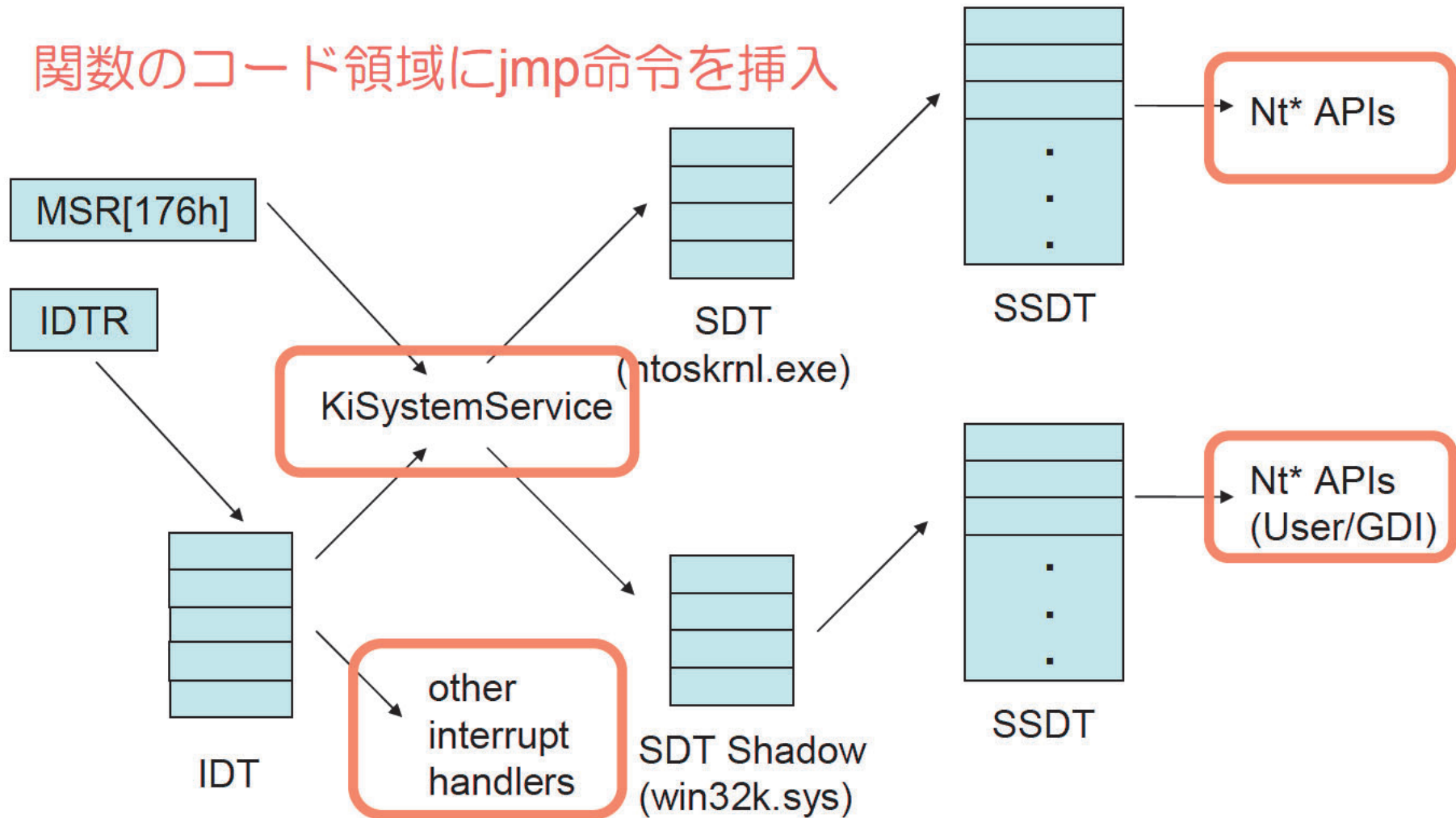
Type I – Windowsにおけるフッキングポイント(2/3)

静的な関数ポインタ、
アドレステーブルを改竄



Type I – Windowsにおけるフッキングポイント(3/3)

関数のコード領域にjmp命令を挿入

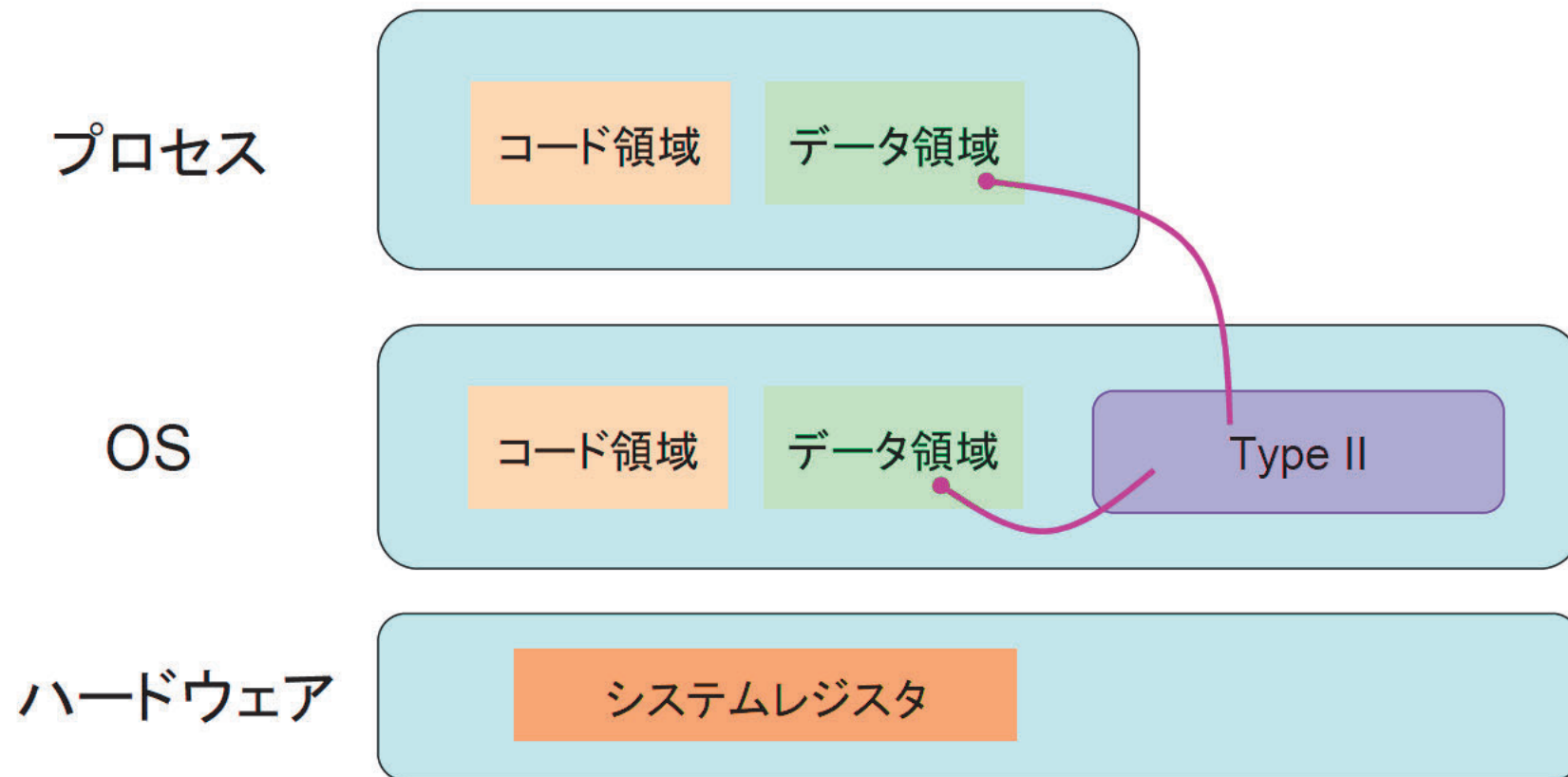




Type Iとその対策

- ・ 検出は容易
- ・ 多くの検出ツール、対策が存在する
 - 3rd partyの検出ツール
 - PatchGuard (Vista x64 Edition)

Type II



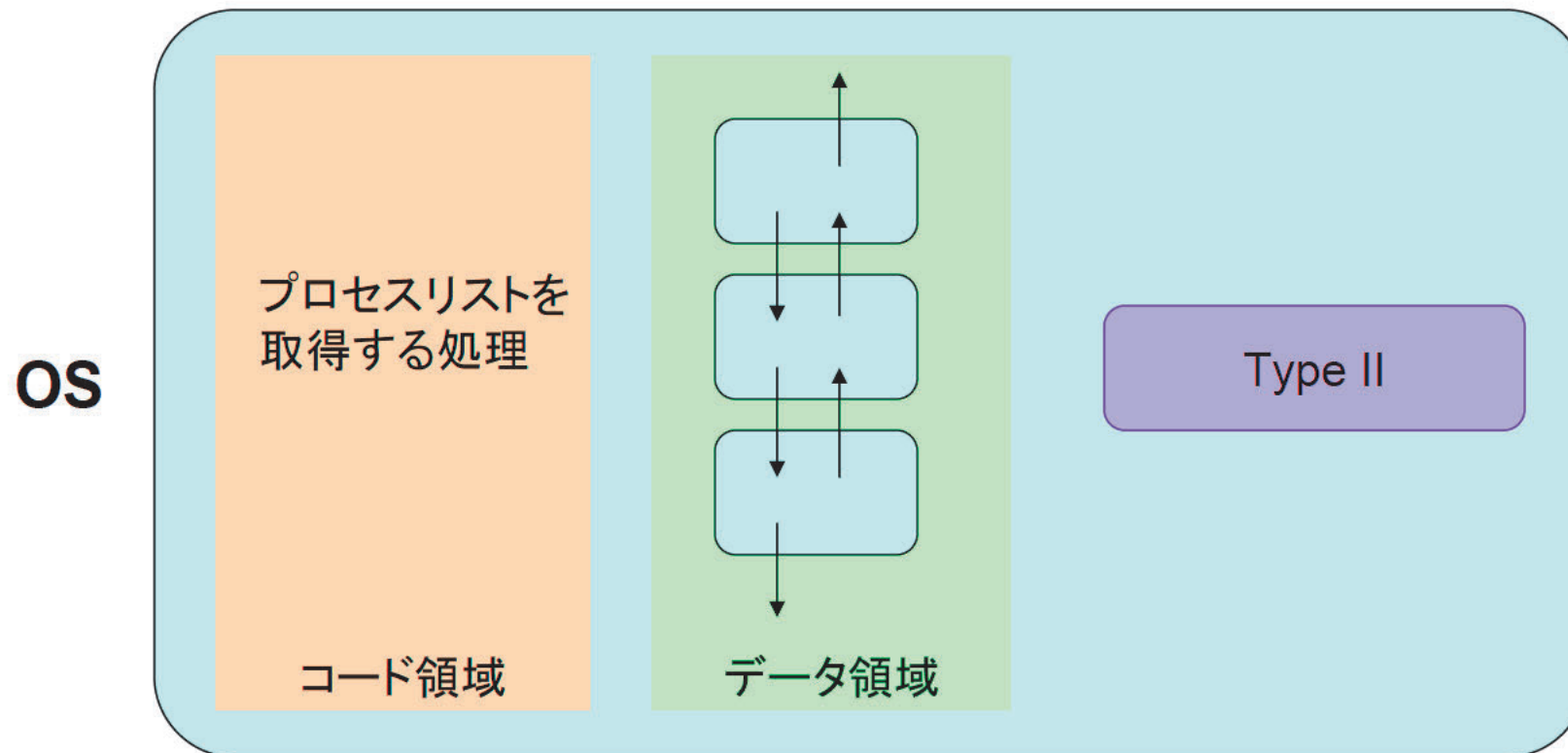


Type II

- データ領域に含まれるデータを改ざん
- DKOM (Direct Kernel Object Manipulation)
Jamie Butler, <http://www.blackhat.com/presentations/win-usa-04/bh-win-04-butler.pdf>
- KOH (Kernel Object Hooking)
Greg Hoglund, <http://www.rootkit.com/newsread.php?newsid=501>

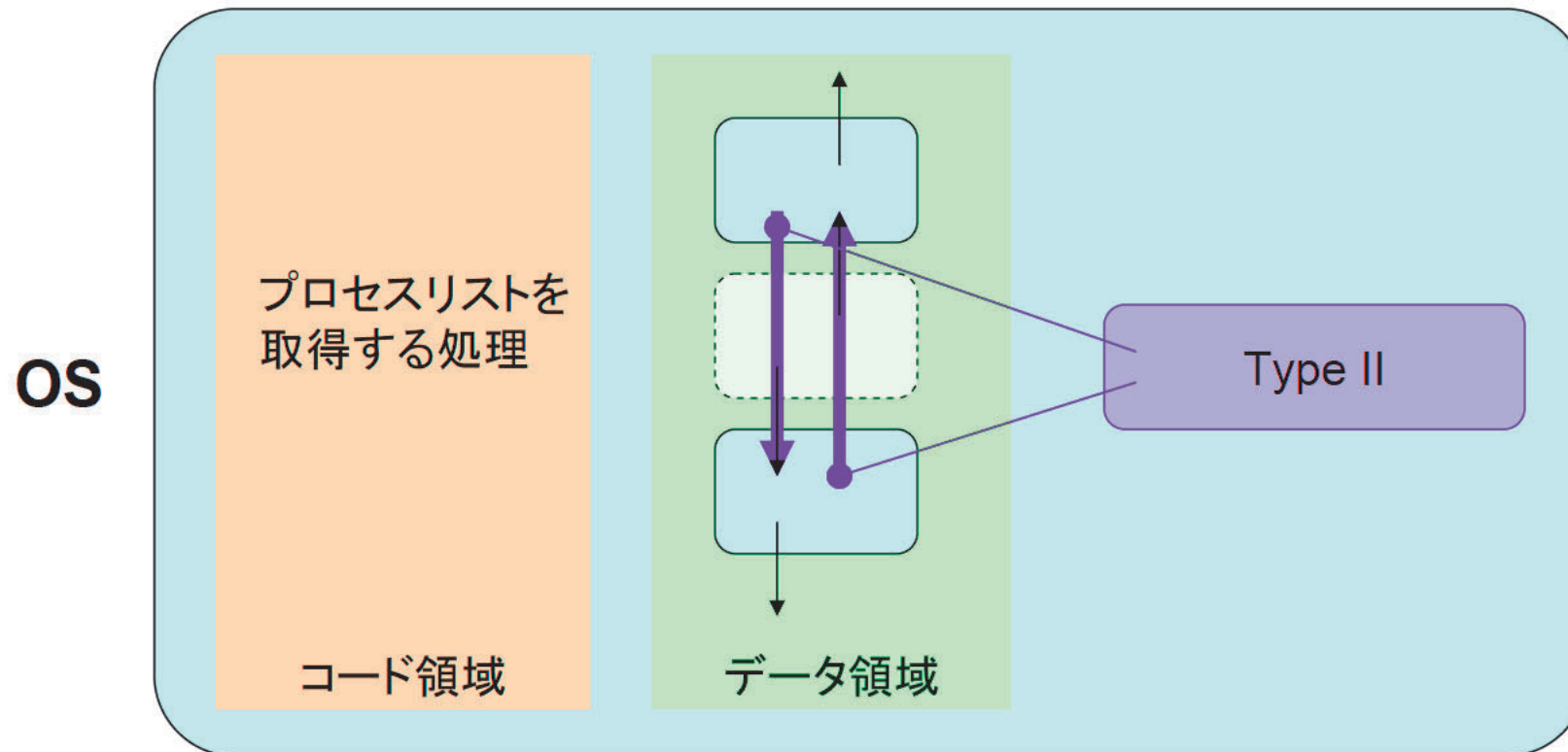
DKOM (Direct Kernel Object Manipulation)

- データ領域に含まれるプロセスリスト、プロセストークン等のデータを改竄



DKOM (Direct Kernel Object Manipulation)

- データ領域に含まれるプロセスリスト、プロセストークン等のデータを改竄





DKOM (Direct Kernel Object Manipulation)

- ・ 限定的な機能しか実装することができない
 - 実現可能な機能は、対象のデータを取り扱う処理の実装に依存する
- ・ プロセスエントリをプロセスリストから除外すると、
 - スケジューリングされない(実行されない)？
 - スレッドベースのスケジューリング
 - スケジューラの実装に依存

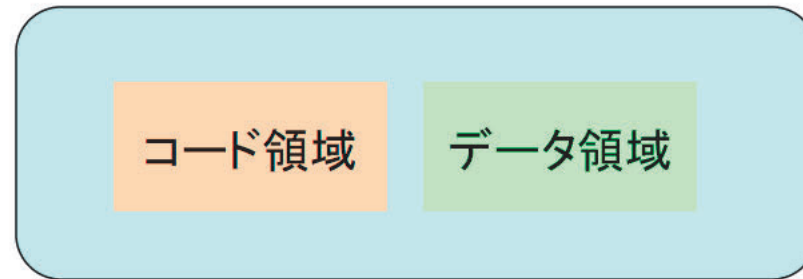


KOH(Kernel Object Hooking)

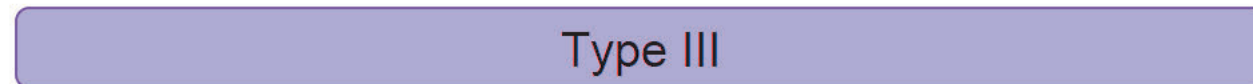
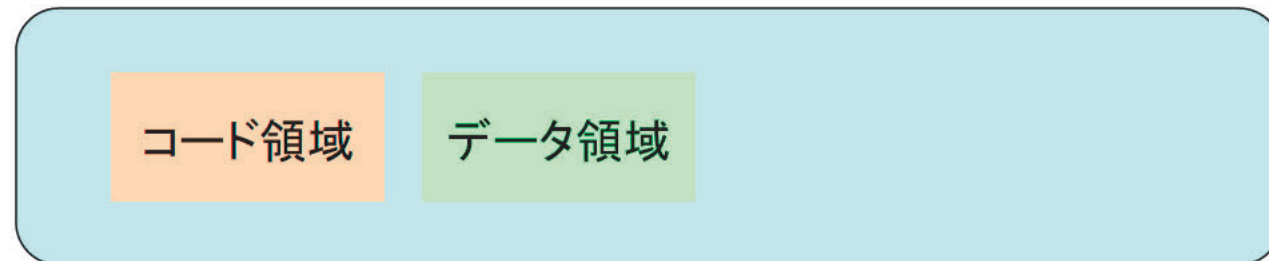
- ・ データ領域に含まれる関数ポインタを改竄
- ・ カーネル内には、IDT、SDT、SSDT以外にも無数の関数ポインタが存在する
- ・ KOHによるフッキングを検出するためには、カーネル空間に存在する全ての関数ポインタを把握する必要がある
 - ルートキット開発側と検出側のイタチごっこ

Type III

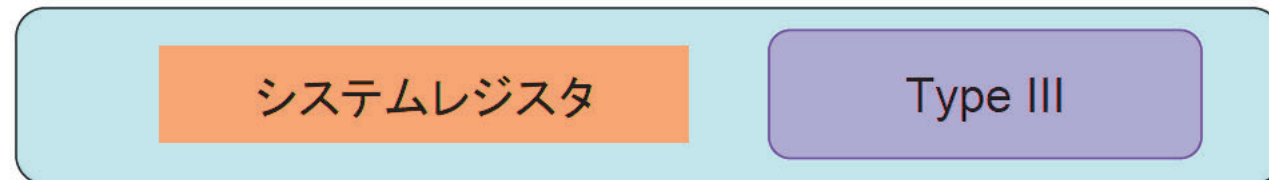
プロセス



OS



ハードウェア





Type III(1/2)

- システム(OS)の外部で動作する
 - 仮想化技術の悪用
 - ファームウェアルートキットやSMMルートキット
- BluePill
 - ハードウェアの仮想化支援機能を悪用し、悪意のハイパーバイザーをインストール
 - (ゲスト)OSのリソースは改ざんしない
 - 最新のBPは、Intel VTおよびAMD-vの両方をサポート

Joanna Rutkowska, Alexander Tereshkin, <http://bluepillproject.org>



Type III(2/2)

- Vitriol
 - Intel VTを悪用するType IIIルートキット
Dino Dai Zovi, Black Hat USA 2006
- VMM Rootkit Framework
 - Vitriol同様、Intel VTを悪用するType IIIルートキット
 - コンセプトコードのため非常にプリミティブな実装
 - VMMの動作、実装を学ぶために最適
 - Shawn Embleton,*
<http://www.rootkit.com/newsread.php?newsid=758>



ケーススタディ: Storm Worm

- ・ 2007年に出回り始めたP2Pボット
- ・ スпамメールに自身の実行ファイルを添付して送信
- ・ スпамメールのサブジェクトに実際に発生した事件を利用
 - “230 dead as storm batters Europe”
 - “Happy New Year 2008”
 - etc.
- ・ 一部の亜種は、AV製品から逃れるためにルートキット機能を備えている



Storm Worm – “Happy New Year 2008”

実行ファイル実行すると、

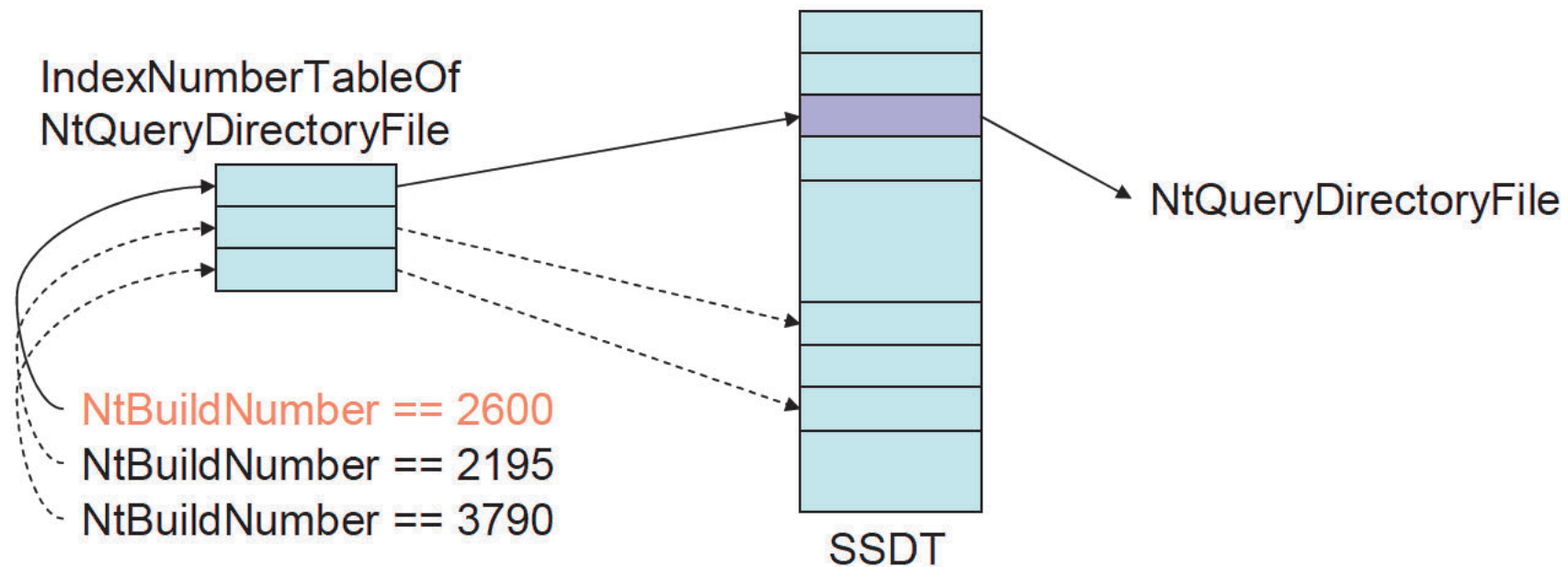
1. システムファイルを生成、カーネルにロード

2. ドライバによる悪意の動作
 - ・ ルートキット機能
SSDT、IRPフッキングを利用したファイル、レジストリ、接続情報の隠ぺい
 - ・ コードインジェクション機能
悪意のコード(本体部分)をユーザーモードプロセスに注入

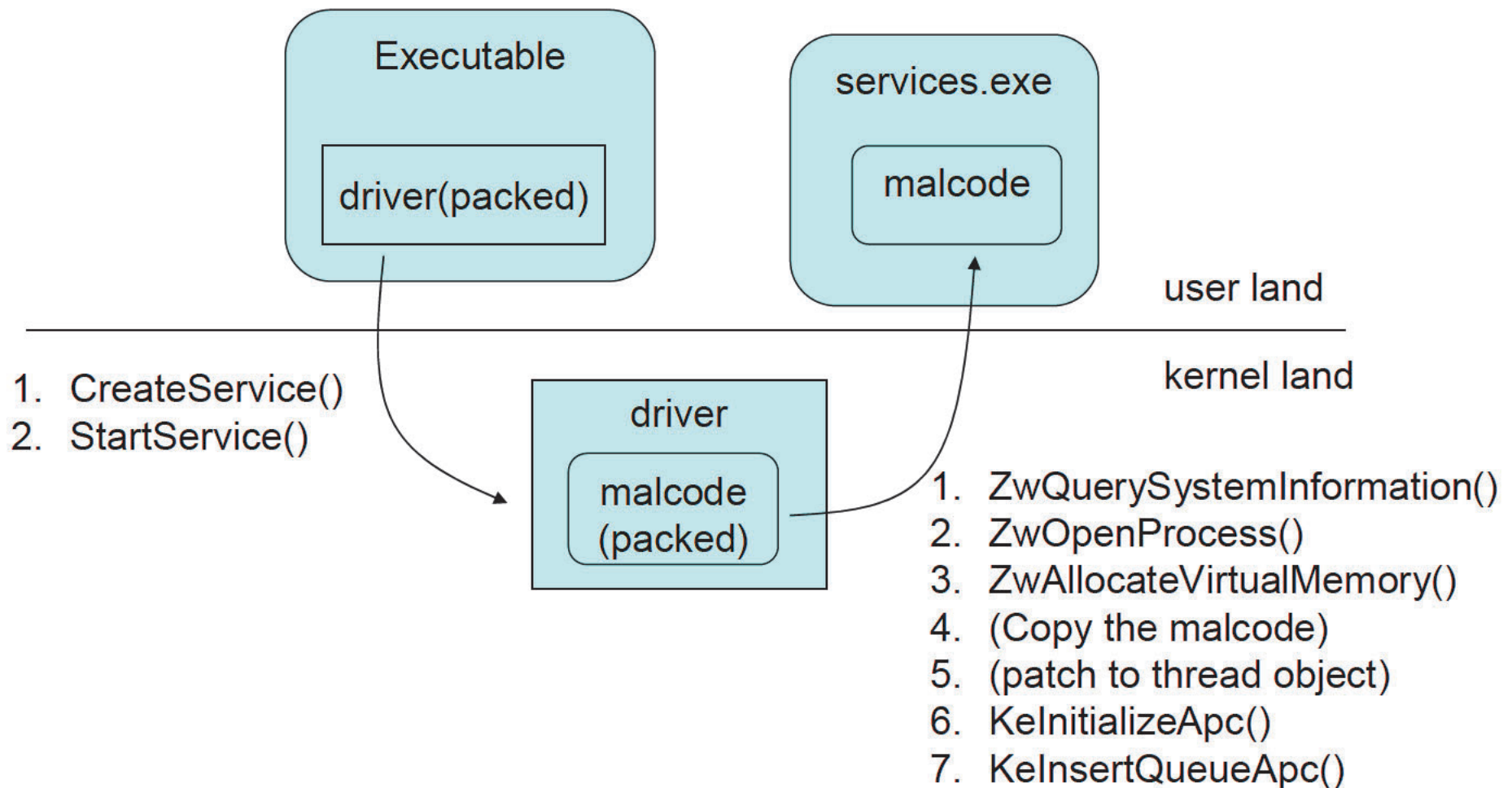
3. P2P通信を開始

ルートキット機能

- ・ 以下の3つのNative APIをフック(SSDTフッキング)
 - NtQueryDirectoryFile, NtEnumerateKey, NtEnumerateValueKey
- ・ APIのSSDTにおけるインデックス番号は、NtBuildNumberによって異なる
- ・ 2195(2k), 2600(XP) and 3790(2k3)の3つをサポート



コードインジェクション機能



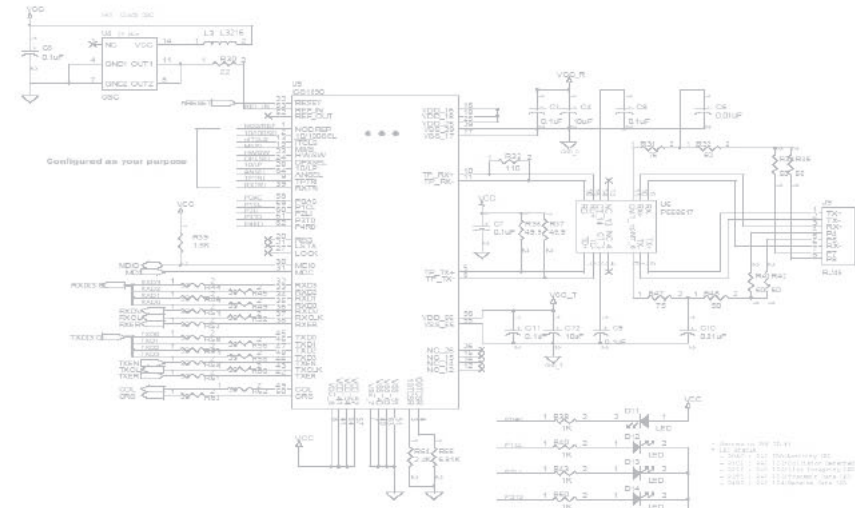


Chapter 2

仮想化技術の概要

```
int packet_analysis(GDDCONFIG *gdd, unsigned char *packet, unsigned long length)
```

```
struct ip      *ip_header; /* IP header */
struct tcphdr *tcp_header; /* TCP header */
struct tcpopt *tcp_opt; /* TCP options */
struct in_addr *src_addr; /* Source IP address */
struct in_addr *dst_addr; /* Destination IP address */
unsigned short sourcePort; /* Source Port */
unsigned short destPort; /* Destination Port */
unsigned long len_data; /* Length of data part */
unsigned long iph_len; /* Length of IP header */
unsigned long teph_len; /* Length of TCP header */
...
static char datestr[512]; /* Buffer to store datetime */
time_t timeval;
struct tm *tmesp=NULL;
char *timesp=NULL;
char *tc;
...
/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_p!=IPPROTO_TCP)
[ ] ip_header->ip_v!=4) return(0);
iph_len = ((unsigned long) (ip_header->ip_hl))+4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) [
return(0);
]
...
/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcph_len = ((unsigned long) (tcp_header->th_off))+4;
tcp_data = (char *) tcp_header+tcph_len;
if (tcph_len<MINSIZE_TCP) return(0);
...
/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcph_len)>0)
return(0);
len_data = (unsigned long)ntohs(ip_header->ip_len)
-iph_len-tcph_len;
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_src), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
memcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
```



00001870	FF 15 F0 11 00 01 E9 00 03 00 00 E8 78 10 00 00	3. 離
00001880	33 F8 56 E8 B1 F0 FF 85 00 0F 84 B7 03 00 00	Vj. 51
00001890	56 6A 02 FF 35 60 80 00 01 FF 35 D0 87 00 01 FF	2. ... u.h. 5
000018A0	15 0C 11 00 01 85 00 75 10 68 10 10 00 00 FF 35	P. 50 52
000018B0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	52
000018C0	FF 15 04 12 00 01 FF 35 00 87 00 01 FF 15 2C 12	5t. ... 誠
000018D0	00 01 FF 35 04 88 00 01 FF 15 68 10 00 01 E9 64	w 山
000018E0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	3. 95 離 t
000018F0	11 0F 85 16 01 00 00 03 F6 39 35 E8 87 00 01 74	= (V. 5V 引
00001900	22 8B 30 28 12 00 01 56 FF D7 56 FF D7 68 00 10	5P. 5. ... 意
00001910	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 70	... 意
00001920	02 00 00 6A 01 E8 0F FG FF FF E9 1A 00 00 00 8B	... 離
00001930	70 14 88 11 01 00 00 2B F0 0F 87 85 00 00 2B	... 離
00001940	F0 0F 84 16 02 00 00 83 FE 10 0F 85 8D 00 00 00	3. 9u. t. ...
00001950	33 F8 39 75 10 74 2F A1 EC 87 00 01 8B 00 F0 87	... 離
00001960	00 01 38 06 75 08 38 CE 0F 84 D9 02 00 00 8B 3D	... 離
00001970	14 12 00 01 51 50 68 B1 00 00 00 FF 35 04 87 00	... 離
00001980	01 E9 56 01 00 00 2B 2D 14 12 00 01 68 F0 87 00	... 離
00001990	01 68 EC 87 00 01 68 80 00 00 FF 35 04 87 00	... 離
000019A0	01 FF D7 A1 EC 87 00 01 8B 00 F0 87 00 01 3B C1	... 離
000019B0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	... 離
000019C0	02 00 00 51 50 E9 07 01 00 00 8B CE 88 12 01 00	... 離
000019D0	00 2B 00 0F 84 30 02 00 00 83 E9 04 0F 84 29 02	... 離
000019E0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 0F	... 離
000019F0	84 E9 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	... 離
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 60	... 離
00001D10	88 09 01 0F 85 EE 00 00 00 2B 45 14 8B 48 0C 8B	... 離
00001D20	01 8B 01 D7 80 01 EA 02 83 E0 01 83 E2 01 F6 01	... 離

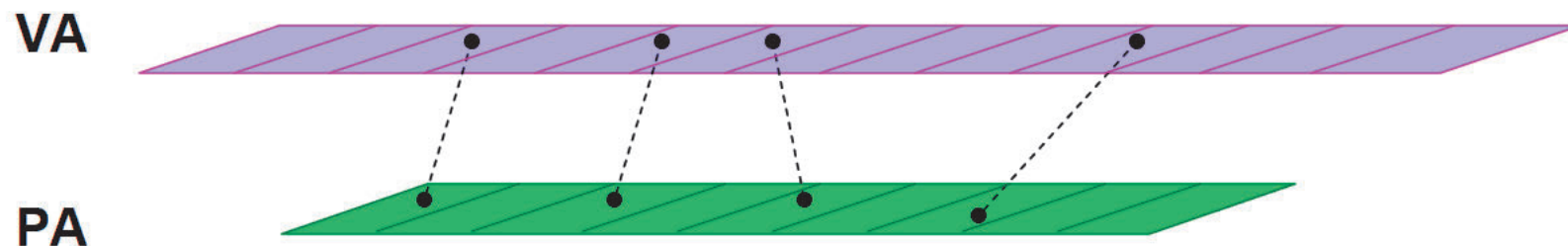


仮想化の必要があるリソース

- ・ 仮想化技術を利用した場合、1つのVMM(仮想マシンモニタ)、1つ以上のVM(仮想マシン)が動作する
 - 従来OSが占有していたリソースを抽象化する必要がある
 - CPU
 - ・ システムレジスタ、制御レジスタ
 - ・ 例外
 - メモリ
 - ・ VMM、各VMごとの独立したメモリ空間
 - ハードウェア
 - ・ 割り込み、I/Oアドレス空間、DMAアクセス、etc.

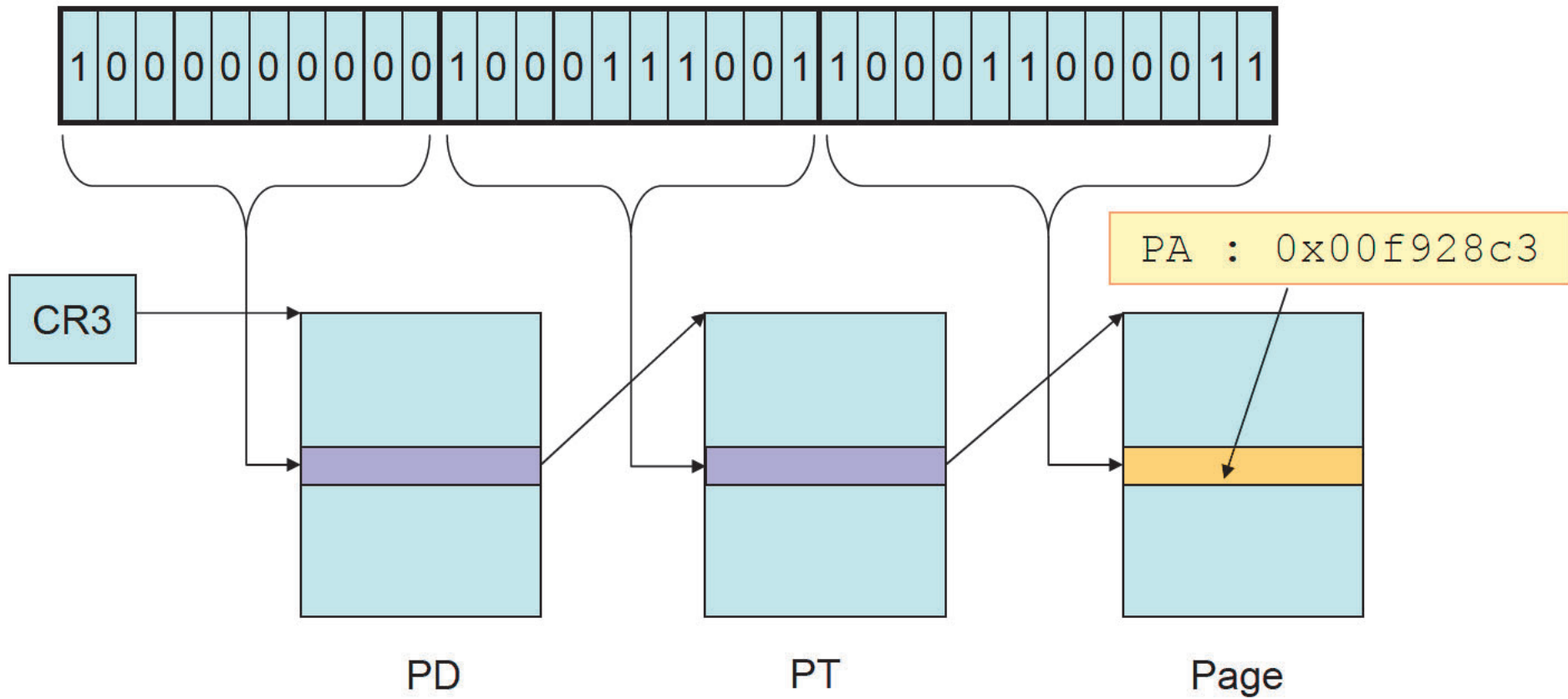
従来の仮想メモリの仕組み

- ・ 32bit環境では、
 - 実際に搭載している物理メモリの容量に限らずOSからは4GBの仮想メモリが見える
 - OS・アプリケーションは、物理アドレス(PA)を意識せず4GBの仮想アドレス(VA)にアクセスする
 - プロセッサが、物理アドレスと仮想アドレスのマッピングを自動的に行う



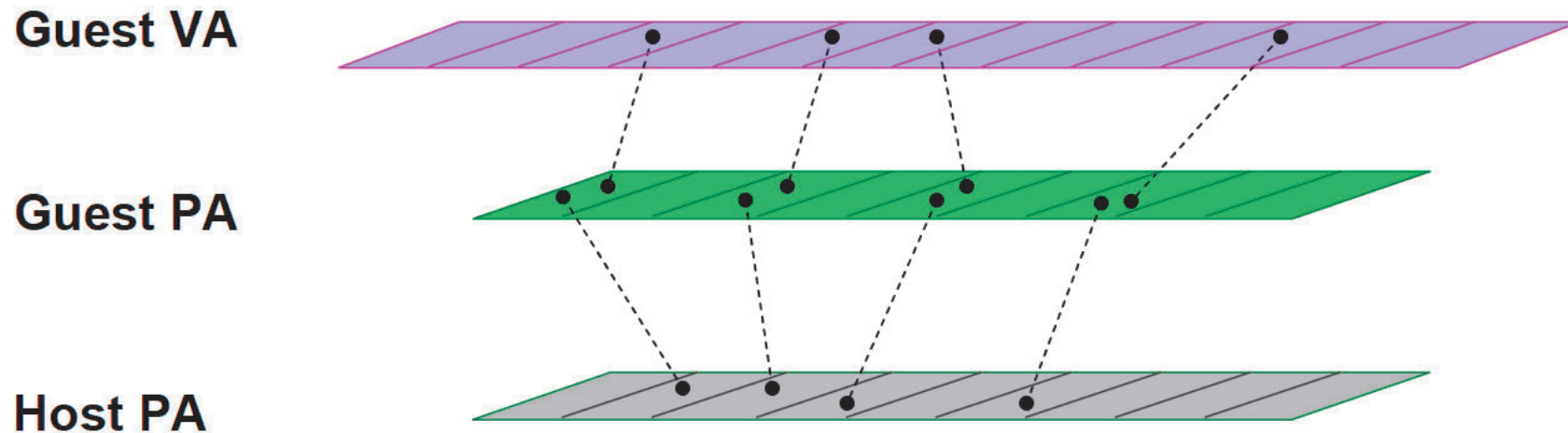
仮想アドレスから物理アドレスへの変換

VA : 0x802398c3

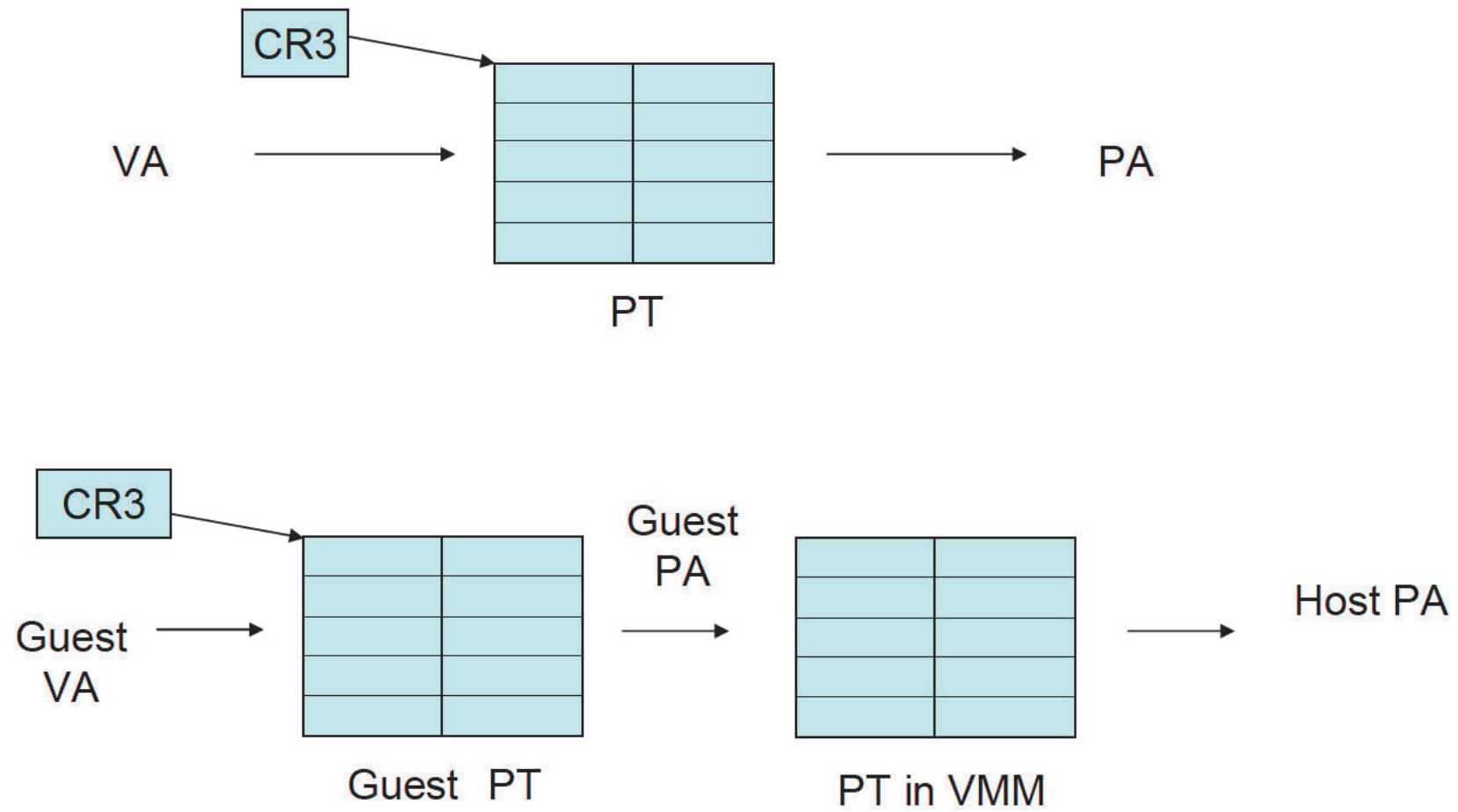


仮想化技術を利用した場合のメモリ仮想化(1/2)

- ・ 各VMが独立したアドレス空間を持つ
- ・ 各VMから見える物理アドレスを更に抽象化する必要がある
 - ゲスト仮想アドレス(Guest VA)、ゲスト物理アドレス(Guest PA)、ホスト物理アドレス(Host PA)



仮想化技術を利用した場合のメモリ仮想化(2/2)



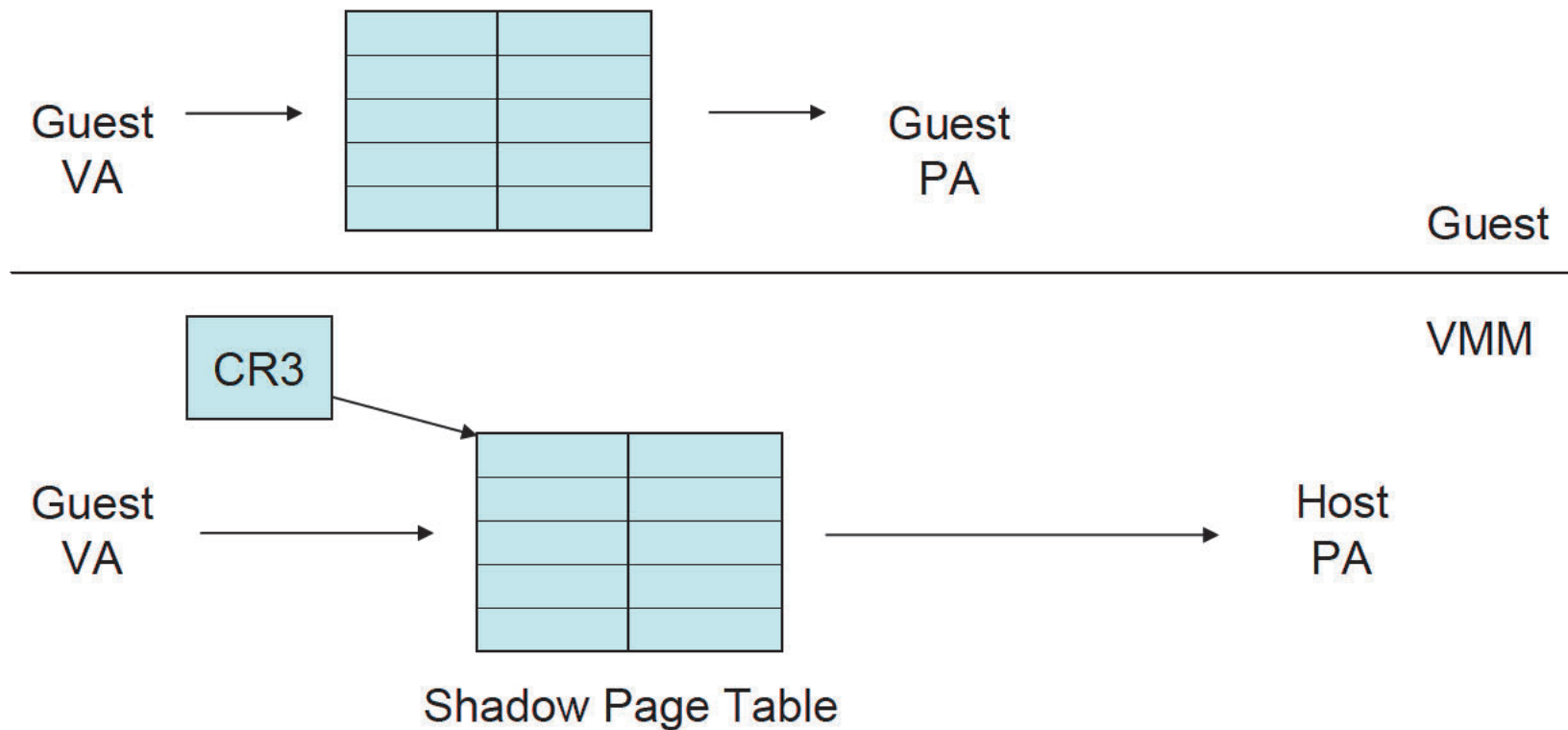


VMMによるアドレス変換

- ・ 現在のプロセッサは、1段階のアドレス変換しかサポートしていない
- ・ VMMは、Guest VA → Guest PA → Host PAの2段階のアドレス変換をソフトウェア的に行う必要がある
 - Shadow Paging
- ・ プロセッサによる2段階アドレス変換のサポート
 - Intel: EPT (Extended Page Table)
 - AMD: NPT (Nested Page Table)

Shadow Paging

- ・ ゲストOSのページフォルト例外をインターセプトし、VMMが適時Shadow Page Table(SPT)を更新



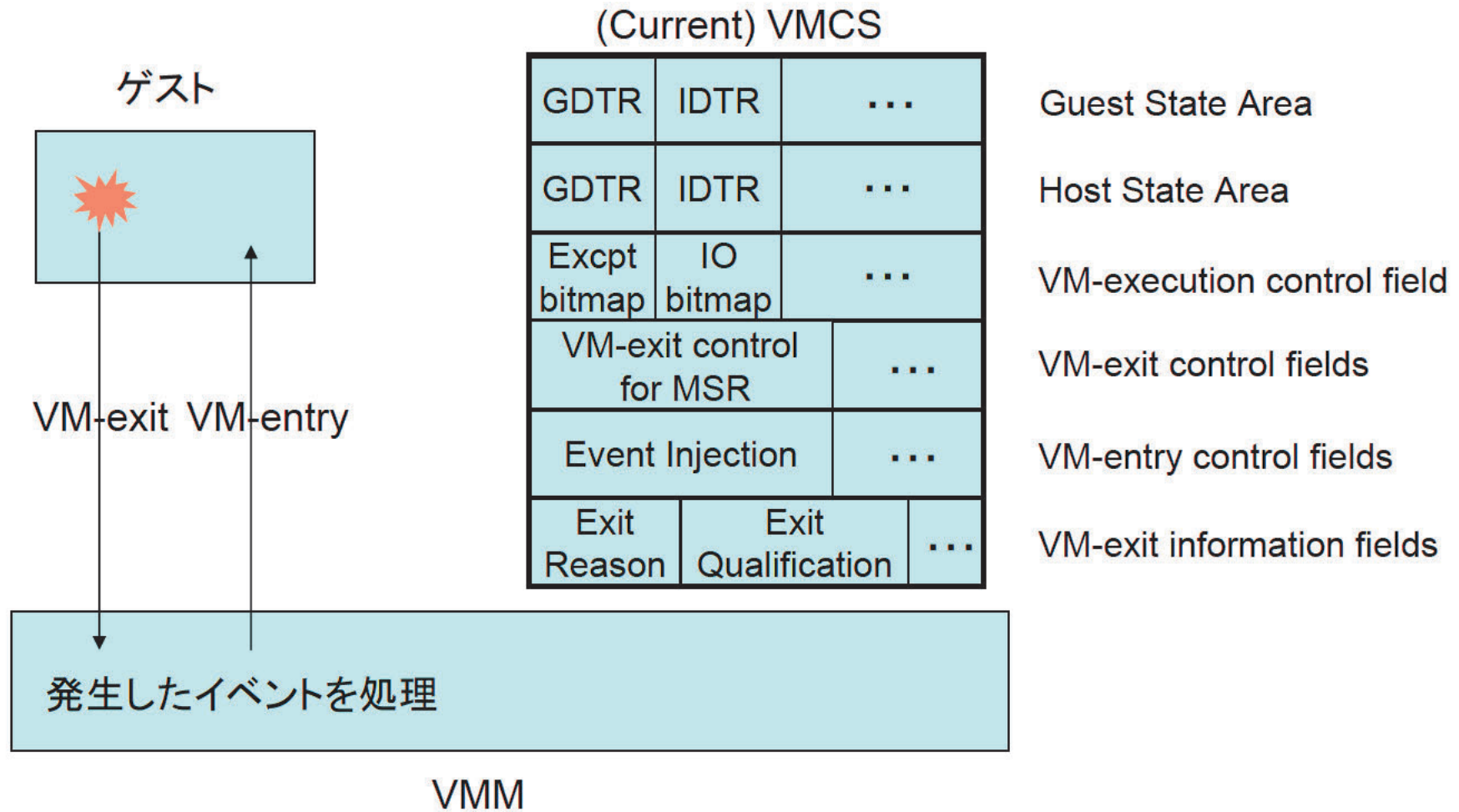


Intel VT

- ・ Intel VTはIntelによる仮想化技術の総称
 - VT-x x86/64向けの仮想化サポート
 - VT-d チップセットによるデバイスの仮想化サポート
 - VT-i Itaniumプロセッサ向けの仮想化サポート

- ・ 主な機能
 - VMXモードのサポート
 - ・ VMX root-operations(ring0-3)
 - ・ VMX non-root-operations(ring0-3)
 - VMCS (Virtual Machine Control Structure)
 - VMX命令セットのサポート
VMXON、VMXOFF、VMXLAUNCH、VMRESUME、VMCALL、etc.

Intel VTの動作メカニズム





主なVM_EXITの発生理由

- 特定の命令の実行
 - CPUID, INVD, INVLPG, RDTSC, RDPMC, HLT, etc.
 - 全てのVMX命令
- I/O命令
 - IN, OUT, etc.
- 例外(#DB、#BP、#PF、etc.)
- 特定のレジスタへのアクセス
 - コントロールレジスタ
 - デバッグレジスタ
 - MSR(モデル固有レジスタ)
- etc



Chapter 3

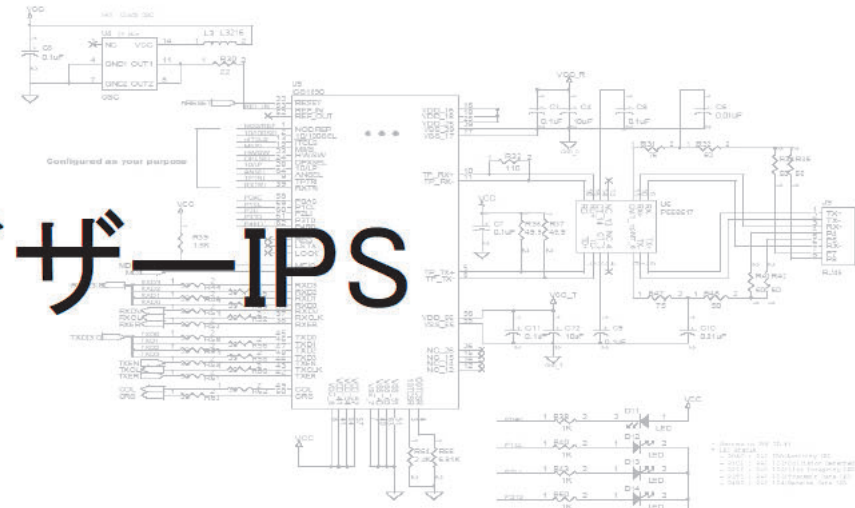
Viton, ハイパーバイザーIPS

```
int packet_analysis(GDDCONFIG *gdd, unsigned char *packet, unsigned long length)
{
    struct ip      *ip_header; /* IP header */
    struct tcphdr *tcp_header; /* TCP header */
    struct tcpopt *top_data; /* TCP data */
    struct in_addr sourceIP; /* Source IP address */
    struct in_addr destIP; /* Destination IP address */
    unsigned short sourcePort; /* Source Port */
    unsigned short destPort; /* Destination Port */
    unsigned long len_data; /* Length of data part */
    unsigned long iph_len; /* Length of IP header */
    unsigned long tcph_len; /* Length of TCP header */
    unsigned long sequence; /* Expected sequence */
    unsigned long nextIndex; /* Index of next port list */
    unsigned int type; /* Log type */
    ConnList *connList; /* Connection table list */
    ConnList *conn; /* Temporary connection list */
    static char datestr[512]; /* Buffer to store datetime */
    time_t timeval;
    struct tm *timep=NULL;
    char *timesp=NULL;
    char *tc;
}

/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_p!=IPPROTO_TCP)
    return(0);
iph_len = ((unsigned long) (ip_header->ihl))*4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
    return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) {
    return(0);
}

/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcph_len = ((unsigned long) (tcp_header->th_off))*4;
tcp_data = (char *) tcp_header+tcph_len;
if (tcph_len<MINSIZE_TCP) return(0);

/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcph_len)>0)
    return(0);
len_data = (unsigned long)ntohs(ip_header->ip_len)
            -iph_len-tcph_len;
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_src), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
memcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
}
```



00001870	FF 15 F0 11 00 01 E9 00 03 00 00 E8 78 10 00 00	3. 離
00001880	33 F8 56 E8 B1 F0 FF 85 00 0F 84 B7 03 00 00	Vj. 51
00001890	56 6A 02 FF 35 60 80 00 01 FF 35 D0 87 00 01 FF	2. u.h. 5
000018A0	15 0C 11 00 01 85 00 75 10 68 10 10 00 00 FF 35	P. 50 52
000018B0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	52
000018C0	FF 15 04 12 00 01 FF 35 00 87 00 01 FF 15 2C 12	5t. X. 誠
000018D0	00 01 FF 35 04 88 00 01 FF 15 68 10 00 01 E9 64	w. 山
000018E0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	3. 95 離 t
000018F0	11 0F 85 16 01 00 00 03 F6 39 35 E8 87 00 01 74	= V. 5V 糸
00001900	22 8B 30 28 12 00 01 56 FF D7 56 FF D7 68 00 10	5P. 5. 基
00001910	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 70	離 離
00001920	02 00 00 6A 01 E8 0F FG FF FF E9 1A 00 00 8B	3. 9u. t. *
00001930	70 14 88 11 01 00 00 2B F0 0F 87 85 00 00 2B	u. 5. 5. 離
00001940	F0 0F 84 16 02 00 00 83 FE 10 0F 85 8D 00 00 00	OP. 離
00001950	33 F8 39 75 10 74 2F A1 EC 87 00 01 8B 00 F0 87	*. <
00001960	00 01 38 06 75 08 3E 0E 0F 84 D9 02 00 00 8B 3D	1.
00001970	14 12 00 01 51 50 68 B1 00 00 00 FF 35 04 87 00	離 離
00001980	01 E9 56 01 00 00 2B 2D 14 12 00 01 68 F0 87 00	h. t. 5t.
00001990	01 68 EC 87 00 01 68 80 00 00 00 FF 35 04 87 00	*. 離
000019A0	01 FF D7 A1 EC 87 00 01 8B 00 F0 87 00 01 3B C1	5.
000019B0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	u. 5. 5. 離
000019C0	02 00 00 51 50 E9 07 01 00 00 8B CE 88 12 01 00	OP. 離
000019D0	00 2B 00 0F 84 30 02 00 00 83 E9 04 0F 84 29 02	*. <
000019E0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 0F	1.
000019F0	84 E9 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	離 離
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 60	*. 離
00001D10	88 09 01 0F 85 EE 00 00 00 2B 45 14 8B 48 0C 8B	*. 離 離
00001D20	C1 8B D1 F7 80 C1 EA 02 83 E0 01 83 E2 01 F6 C1	離



Viton

- ・ OSの外部で動作するIPS
- ・ Proof of Concept、Windows XP SP2、SP3で動作確認
- ・ 主な機能
 - ✓ 指定したメモリ領域に対するメモリパッチングを禁止
 - ✓ システムレジスタの変更を禁止
 - ✓ ゲストOSの活動をモニタリング

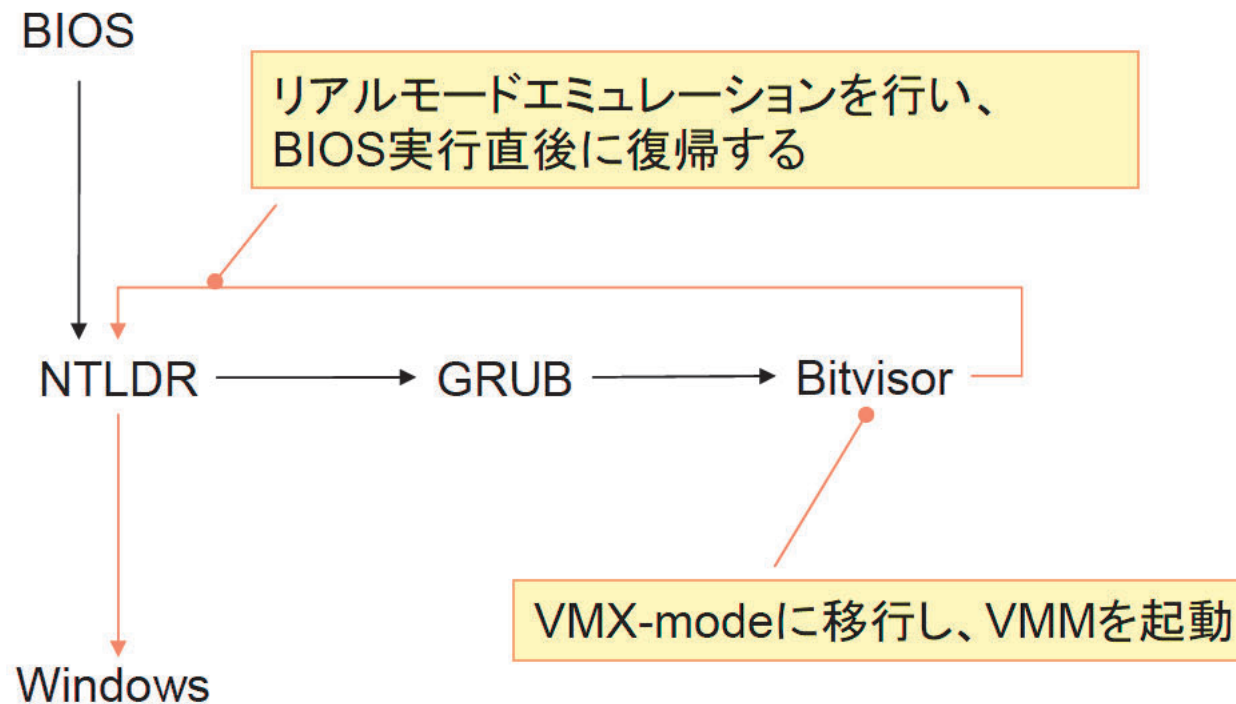
実装には、*BitVisor* を利用



BitVisor

- セキュアVMプロジェクトによって開発されている国産VMMソフトウェア
- 特徴
 - Intel VTを利用した軽量な実装
 - Type I型のVMM(ハイパーバイザー型)
 - VMMにおける32bit・64bitモード対応
 - VMM、ゲストOSにおけるマルチコア・マルチプロセッサ対応
 - WindowsXP/Vista、Linuxを修正無しに実行可能
 - ゲストOSにおけるPAEのサポート
 - リアルモードエミュレーションのサポート
 - etc.

Bitvisorの動作メカニズム



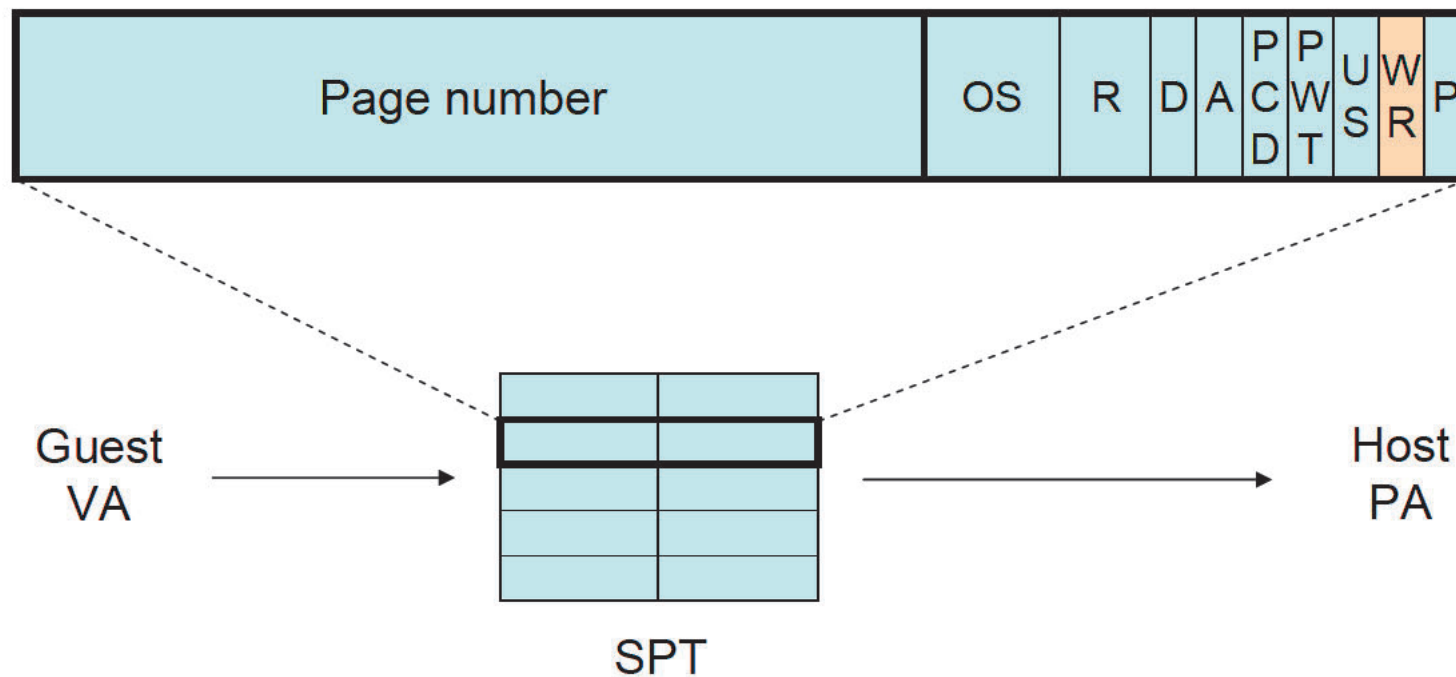


Vitonが保護するリソース

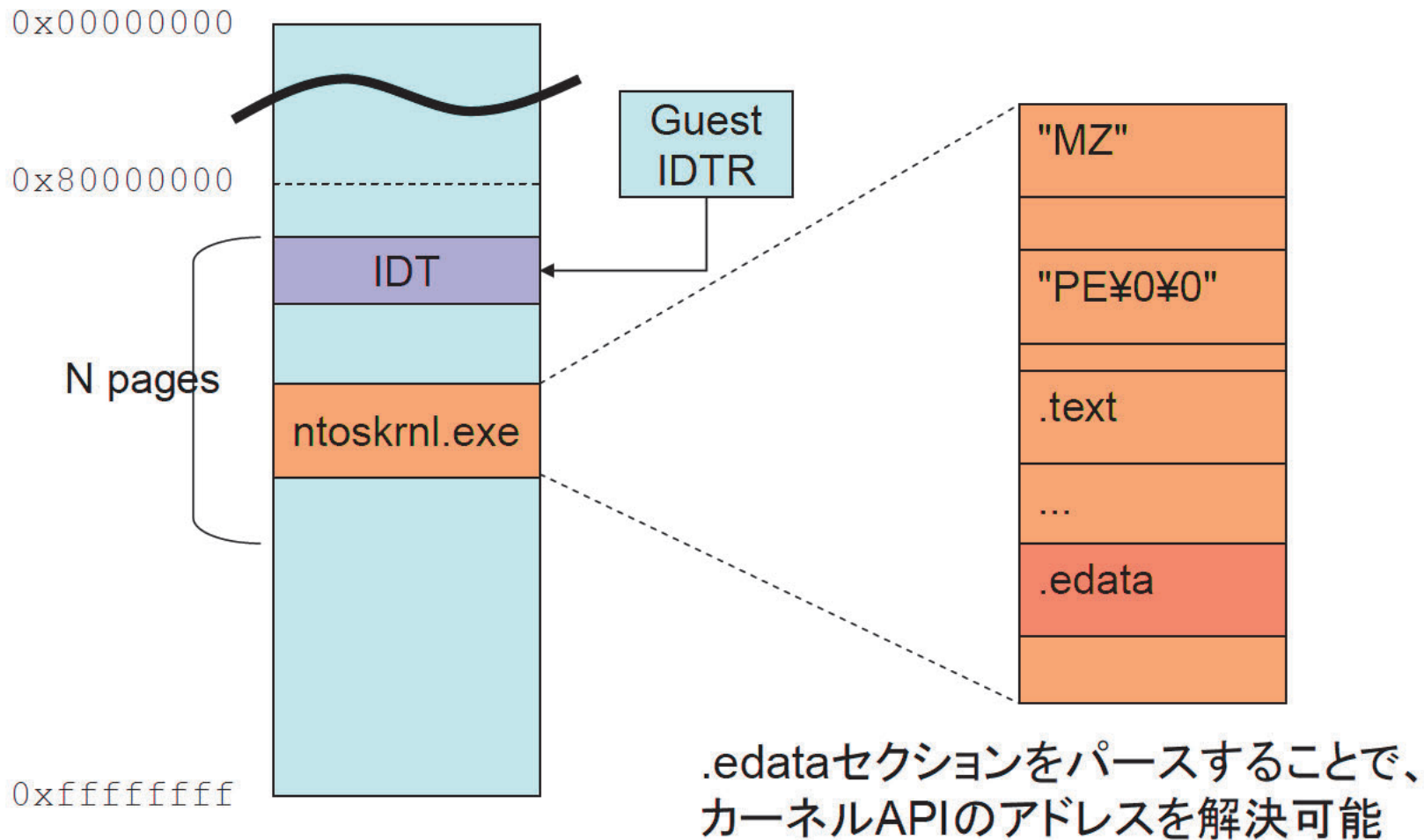
- ・ 命令
 - ゲストOS上での全てのVMX命令を検出・防止
- ・ レジスタ
 - IDTRレジスタの監視
 - 主要なMSRの監視、変更防止
 - CR0.WPビットの監視、変更防止
- ・ メモリ
 - カーネルの全てコード領域
 - IDT
 - SDT
 - SSDT

メモリ保護のメカニズム

- 保護対象の仮想アドレスに対応するPTEのWRビットをクリア
 - CR0.WPが1の場合、ring0の処理でもメモリの書き換えが不可能



ゲストOSのメモリアウトの把握



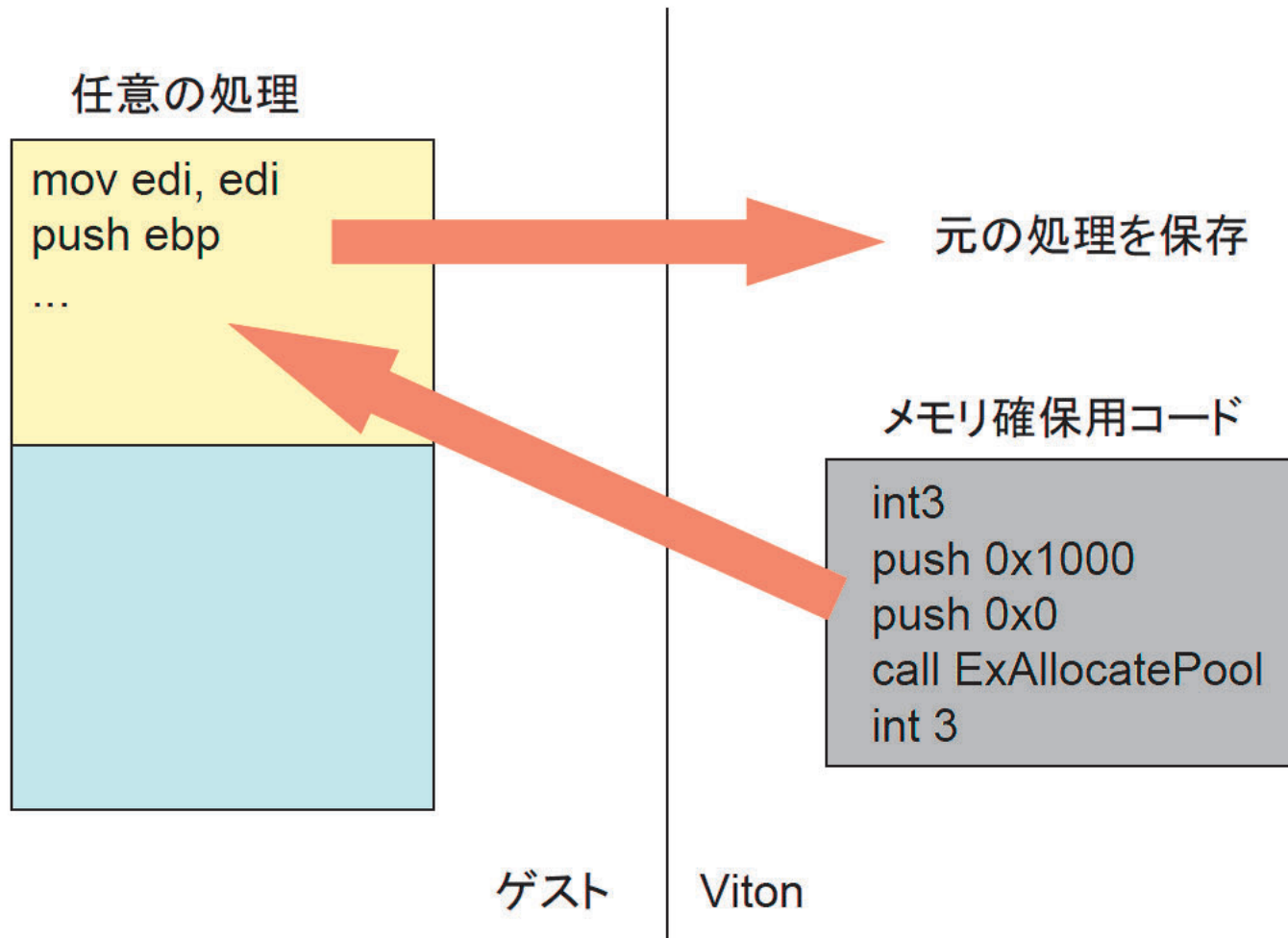


ゲストOSの活動のモニタリング

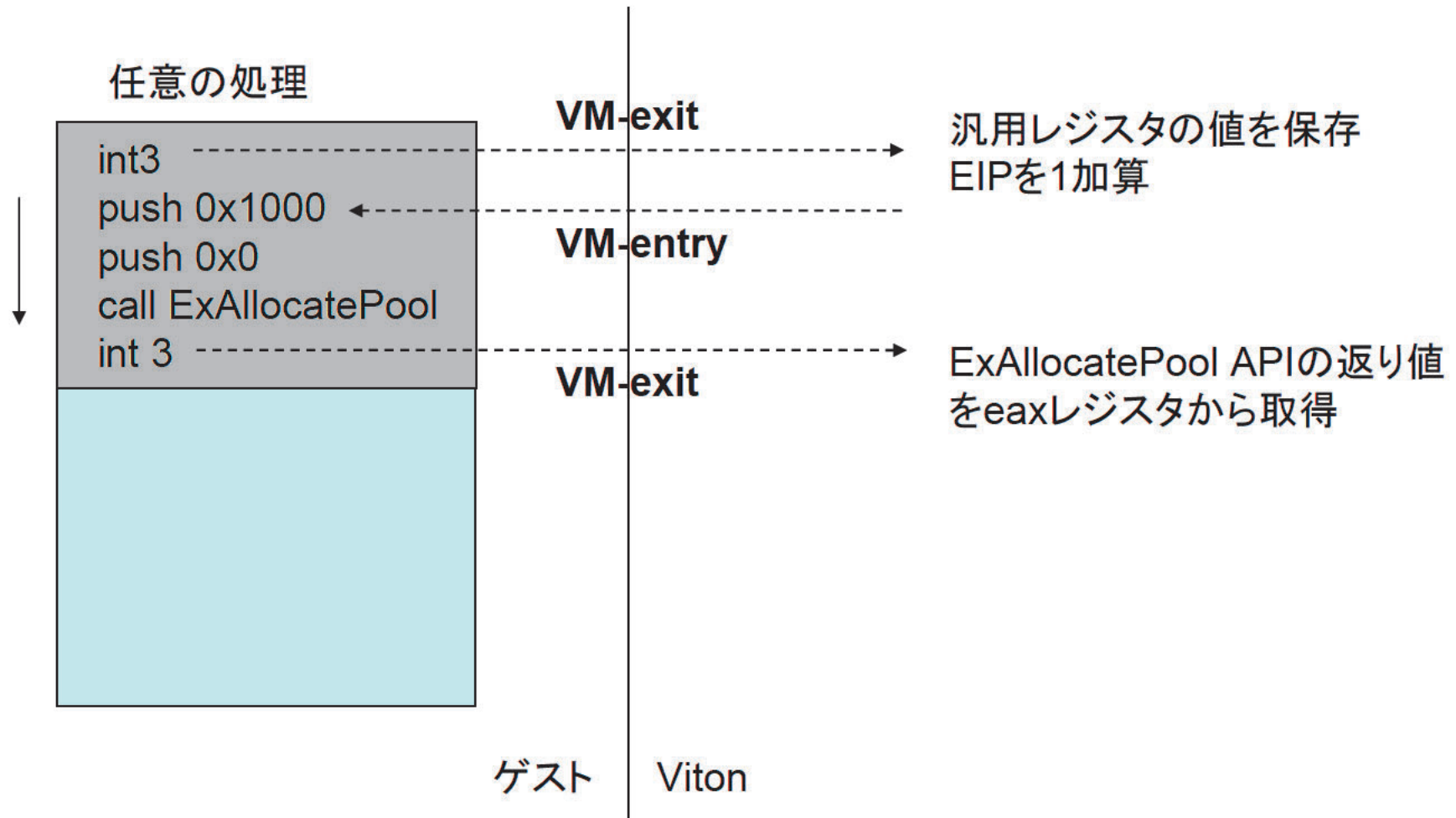
- ・ Vitonを利用した場合、Viton以外からのメモリパッチングを防止することができる

 - ・ VitonからゲストOSへのパッチングは可能
 - ゲストOSの任意の処理をフックし、活動を監視
-
1. フックコード用メモリ領域の確保
 2. フックコードのセットアップ
 3. 対象処理のフック

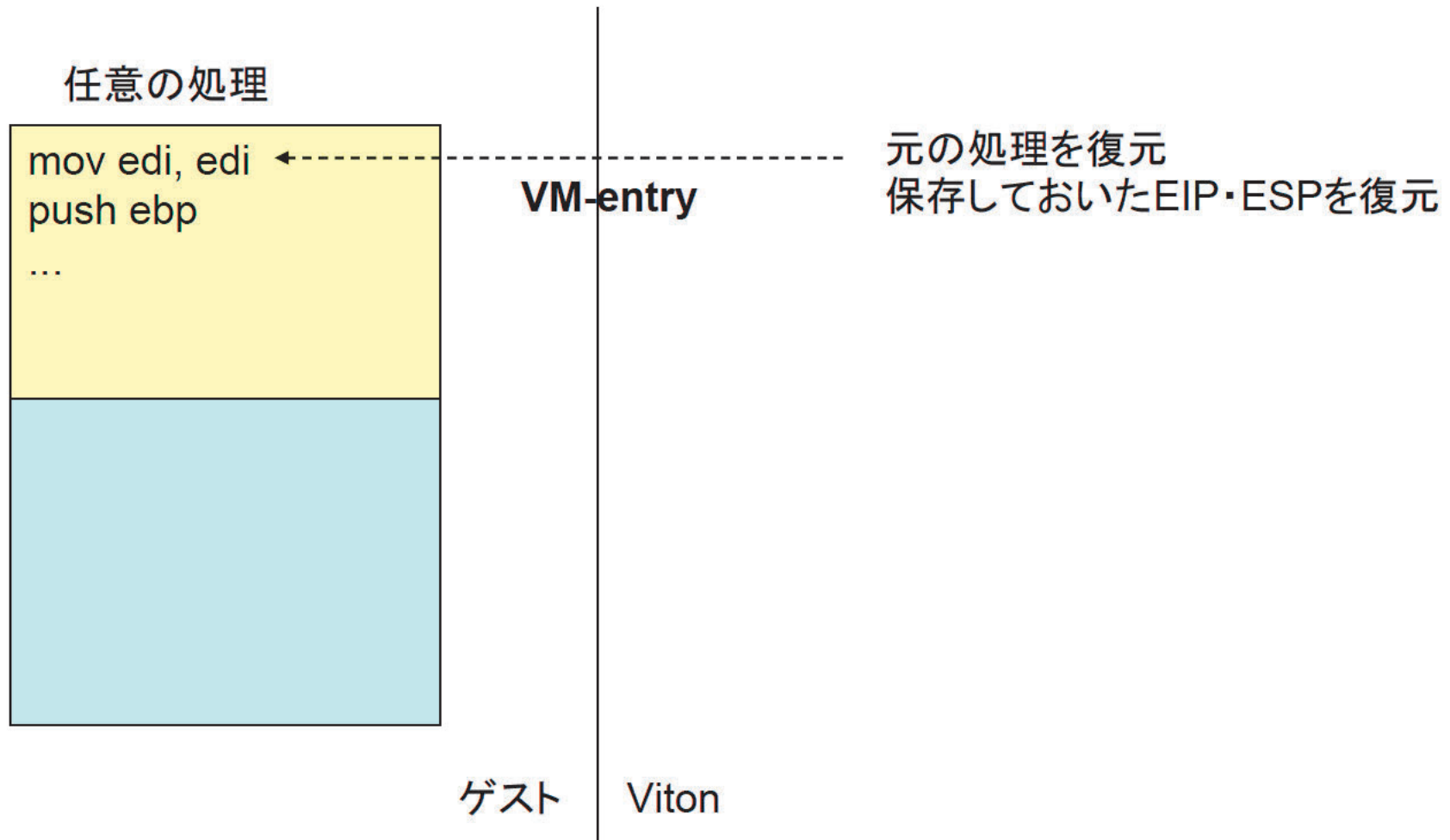
フックコード用メモリ領域の確保(1/3)



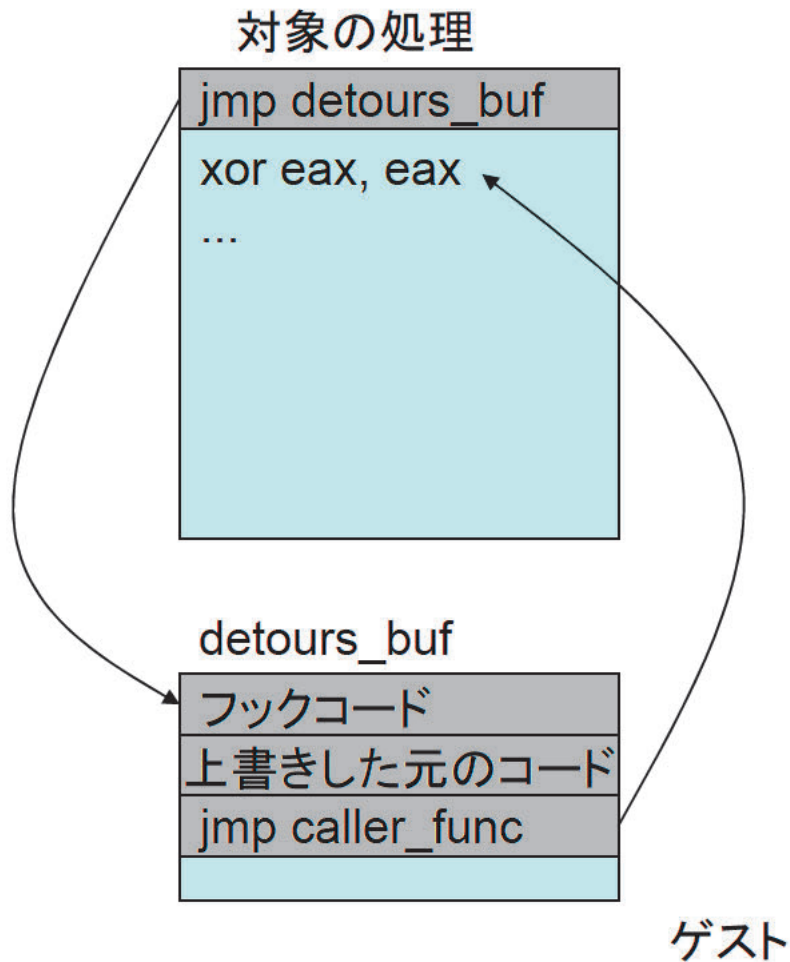
フックコード用メモリ領域の確保(2/3)



フックコード用メモリ領域の確保(3/3)



フックコードのセットアップ & 対象処理のフック



対象の関数が実行されると、

1. detours_bufにジャンプ
2. フックコードが実行される
3. "jmp detours_buf"で上書きした箇所
の元の処理が実行される
4. 呼び出し元の上書きした次の処理へ
ジャンプ



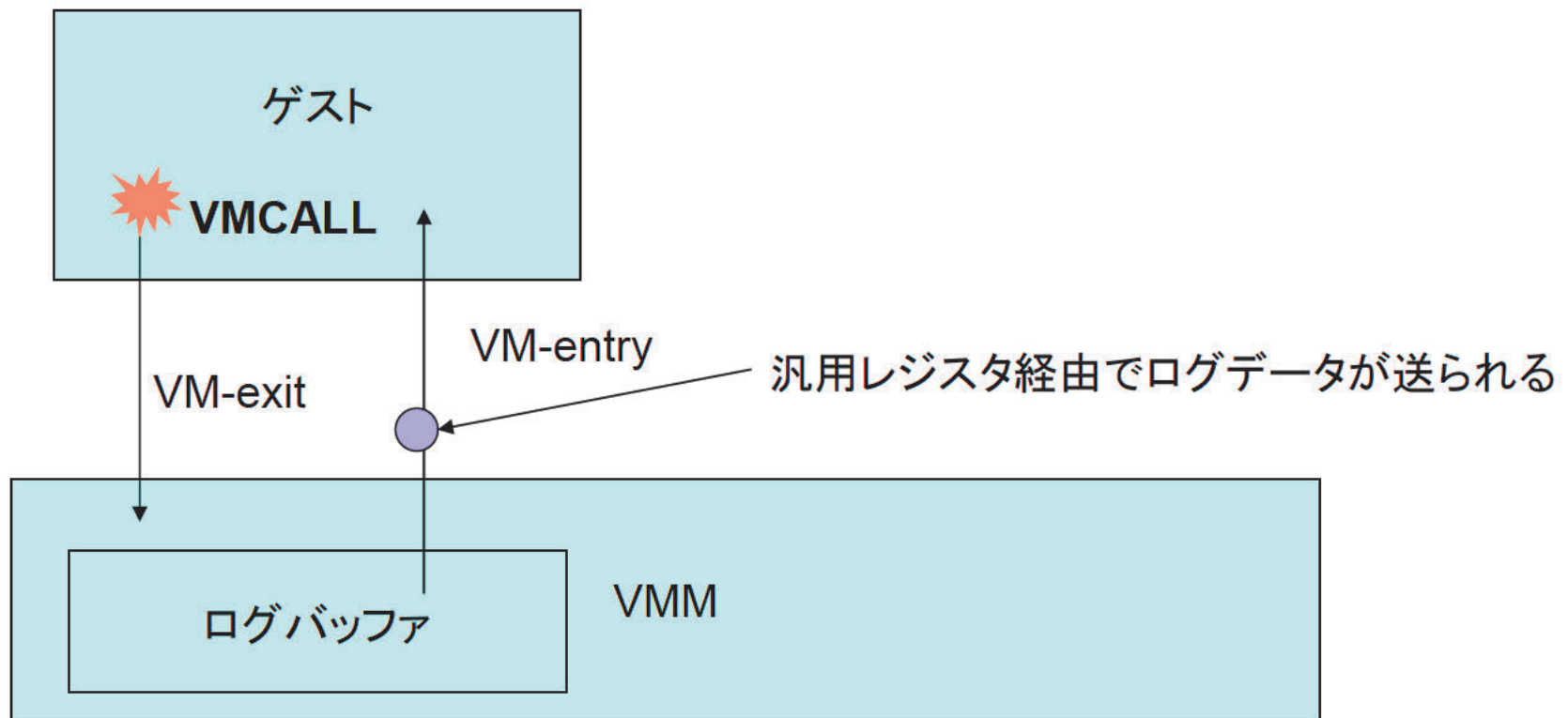
フックコードによる活動の監視

- ・ 下記のAPIをフックすることでゲストOSのリソースを把握可能
 - ZwCreateProcess/ZwTerminateProcess
 - ZwLoadDriver/ZwUnloadDriver
- ・ OS内の情報が改竄されたとしても、Vitonが正しい情報を把握している



デモ

dbgsh (Bitvisor's debugging function)





Viton vs.

- Type I
 - ✓ 「検知」および「防止」が可能
- Type II
 - ✓ DKOM: 活動のモニタリングにより「検知」可能
 - ✓ KOH: 汎用的な対策を行うにはブレークスルーが必要
- Type III
 - ✓ 「検知」および「防止」が可能



まとめ

- ・ 仮想化技術を利用することで効果的なセキュリティ対策を実現することが可能
- ・ 完全ではない
 - 既存ソリューションを保護するための基盤
 - 既存のソリューションとの併用

Thank you!



Fourteenforty Research Institute, Inc.

<http://www.fourteenforty.jp>