



POSITIVE
TECHNOLOGIES

Плагин exploitmon для DRAKVUF. Обнаружение попыток эксплуатации в ядре с помощью PT Sandbox

Павел Максютин
Эксперт PT ESC

Алексей Вишняков
Эксперт PT ESC



ptsecurity.com

Содержание



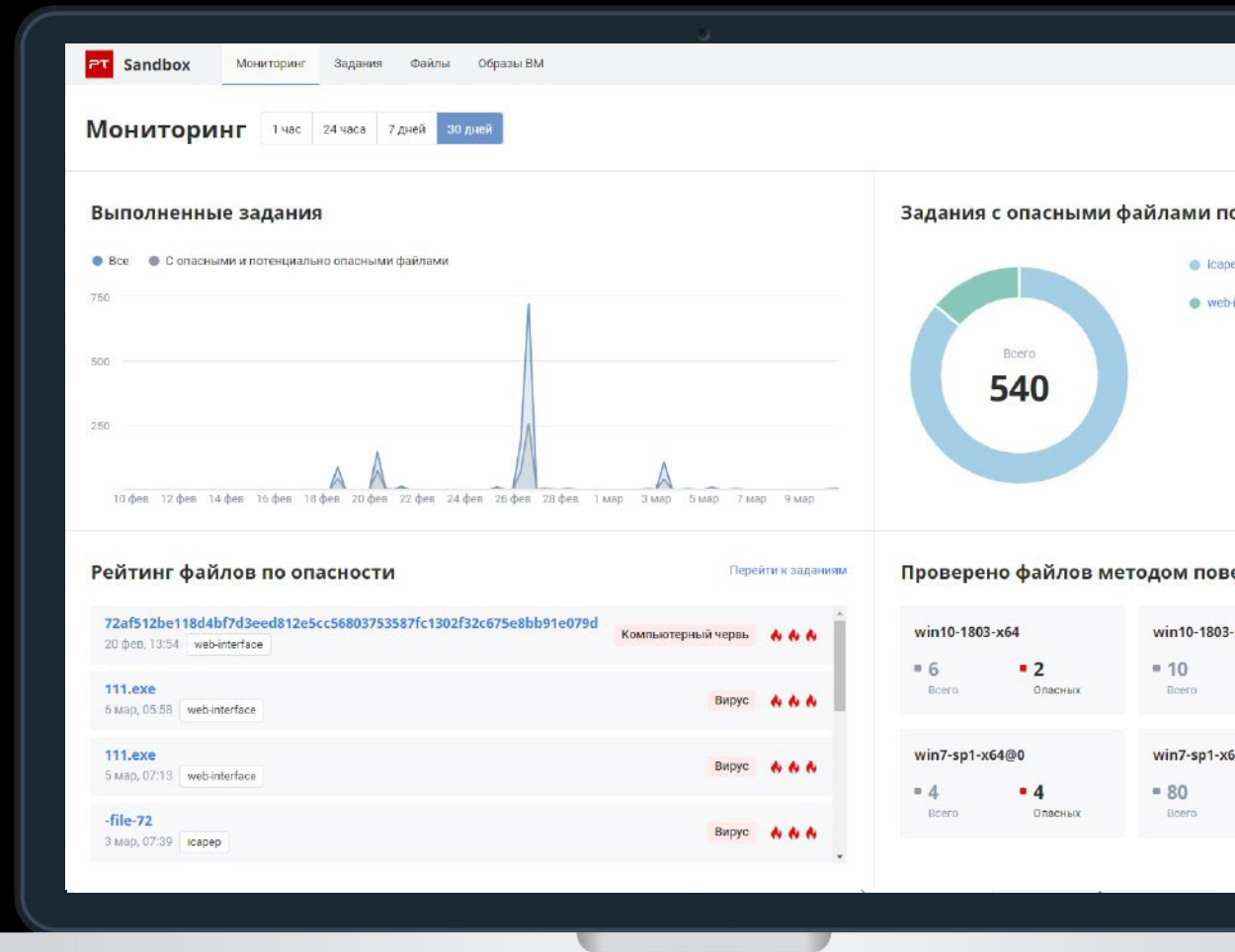
- Коротко о PT Sandbox
- Основные векторы эксплуатации в ядре
- DRAKVUF
- Плагин exploitmon
- Обнаружение эксплойтов повышения привилегий
- Demo
- Заключение

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

The logo consists of the letters 'PT' in a stylized, bold, sans-serif font, enclosed within a white square.

- Коротко о PT Sandbox
- **Основные векторы эксплуатации в ядре**
- DRAKVUF
- Плагин exploitmon
- Обнаружение эксплойтов повышения привилегий
- Demo
- Заключение

Основные векторы эксплуатации в ядре

РТ

03.03.2021 АВТОР: MOVAXBX

WINDOWS KERNEL ZERO-DAY EXPLOIT (CVE-2021-1732) IS USED BY BITTER APT IN TARGETED ATTACK

Original text

Background

In December 2020, DBAPPSecurity Threat Intelligence Center found a new component of **BITTER APT**. Further analysis into this component led us to uncover a zero-day vulnerability in win32kfull.sys. The origin in-the-wild sample was designed to target newest Windows10 1909 64-bits operating system at that time. The vulnerability also affects and could be exploited on the latest Windows10 20H2 64-bits operating system. We reported this vulnerability to MSRC, and it is fixed as **CVE-2021-1732** in the February 2021 Security Update.

Модификация HalDispatchTable

- Таблица указателей функций для HAL
- Находится в ядре, в секции с правами на чтение и запись
- Частично перезаписывается самим HAL при старте системы
- Популярная цель при эксплуатации ядра ОС

Модификация HalDispatchTable

Offset (x86)	Offset (x64)	Definition
0x00	0x00	ULONG Version;
0x04	0x08	NTSTATUS (*HalQuerySystemInformation) (HAL_QUERY_INFORMATION_CLASS, ULONG, PVOID, ULONG *);
0x08	0x10	NTSTATUS (*HalSetSystemInformation) (HAL_SET_INFORMATION_CLASS, ULONG, PVOID);
0x0C		NTSTATUS (*HalQueryBusSlots) (INTERFACE_TYPE, ULONG, ULONG, ULONG *, ULONG *);

Модификация HalDispatchTable

```
kd> uf nt!KeQueryIntervalProfile
```

```
...
```

```
nt!KeQueryIntervalProfile + 0x14:
```

```
82b12cab mov     dword ptr [ebp-10h], eax
```

```
82b12cae lea     eax, [ebp-4]
```

```
82b12cb1 push    eax
```

```
82b12cb2 lea     eax, [ebp-10h]
```

```
82b12cb5 push    eax
```

```
82b12cb6 push    0Ch
```

```
82b12cb8 push    1
```

```
82b12cba call    dword ptr [nt!HalDispatchTable+0x4 (82970434)]
```

```
82b12cc0 test    eax, eax
```

```
82b12cc2 jl     nt!KeQueryIntervalProfile+0x38 (82b12ccf) Branch
```

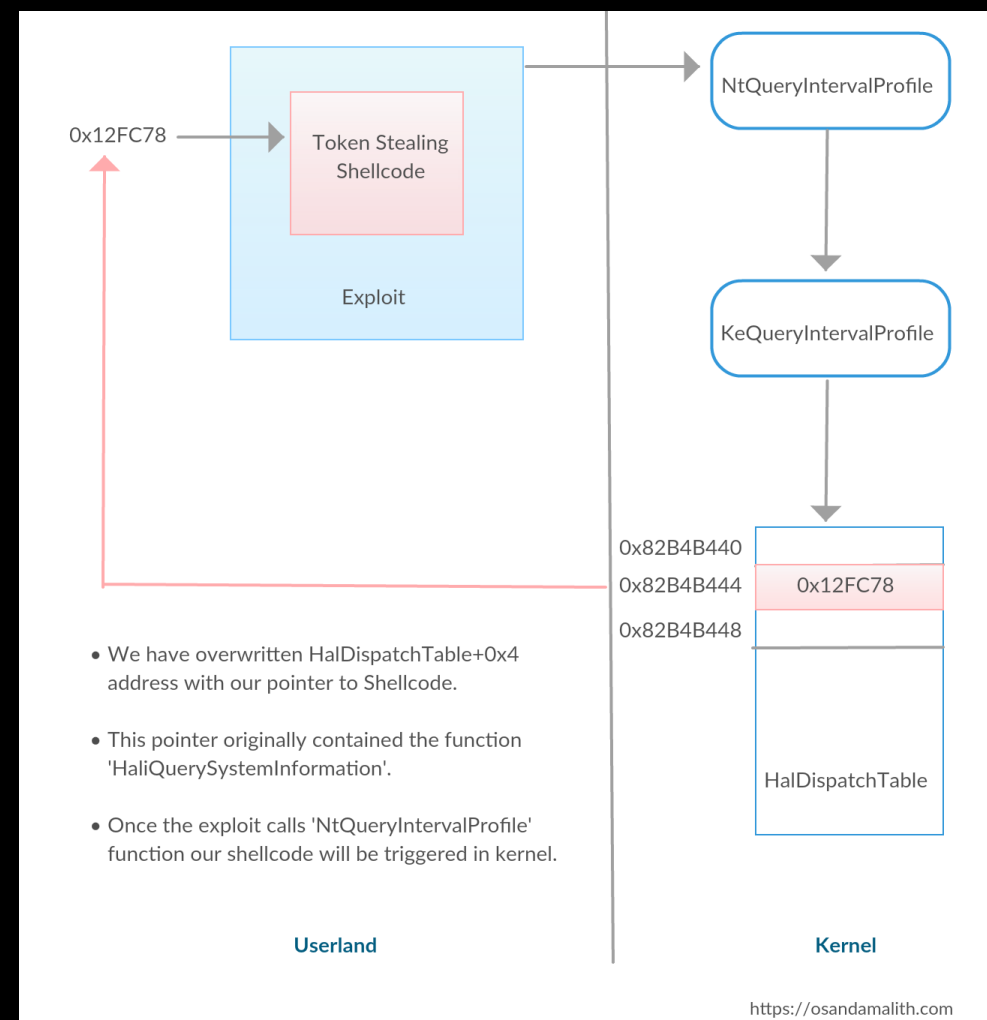
```
...
```


Модификация HalDispatchTable

PT

Шаги эксплуатации:

- Получить примитив на запись, проэксплуатировав уязвимость в ядре ОС
- Переписать указатель в HalDispatchTable на адрес своего шелл-кода
- Вызвать NtQueryIntervalProfile, тем самым исполнив шелл-код в ядре ОС



Модификация токена безопасности

struct EPROCESS

```
typedef struct _EPROCESS
{
    KPROCESS Pcb;
    EX_PUSH_LOCK ProcessLock;
    LARGE_INTEGER CreateTime;
    LARGE_INTEGER ExitTime;
    EX_RUNDOWN_REF RundownProtect;
    PVOID UniqueProcessId;
    LIST_ENTRY ActiveProcessLinks;
    ULONG QuotaUsage[3];
    ULONG QuotaPeak[3];
    ULONG CommitCharge;
    ULONG PeakVirtualSize;
    ULONG VirtualSize;
    LIST_ENTRY SessionProcessLinks;
    PVOID DebugPort;
    union
    {
        PVOID ExceptionPortData;
        ULONG ExceptionPortValue;
        ULONG ExceptionPortState: 3;
    };
    PHANDLE_TABLE ObjectTable;
    EX_FAST_REF Token;
    ULONG WorkingSetPage;
    EX_PUSH_LOCK AddressCreationLock;
```

EPROCESS – структура в ядре ОС Windows, описывающая объект процесса

Модификация токена безопасности



Код кражи токена безопасности от APT28

```
/*
 * StealProcessToken
 *
 * Purpose:
 *
 * Copy system token to current process object.
 *
 */
NTSTATUS NTAPI StealProcessToken(
    VOID
)
{
    NTSTATUS Status;
    PVOID CurrentProcess = NULL;
    PVOID SystemProcess = NULL;

    Status = g_PsLookupProcessByProcessIdPtr((HANDLE)g_OurPID, &CurrentProcess);
    if (NT_SUCCESS(Status)) {
        Status = g_PsLookupProcessByProcessIdPtr((HANDLE)4, &SystemProcess);
        if (NT_SUCCESS(Status)) {
            if (g_EPROCESS_TokenOffset) {
                *(PVOID *)((PBYTE)CurrentProcess + g_EPROCESS_TokenOffset) = *(PVOID *)((PBYTE)SystemProcess + g_EPROCESS_TokenOffset);
            }
        }
    }
    return Status;
}
```

<https://github.com/hfiref0x/CVE-2015-1701/>

Содержание



- Коротко о PT Sandbox
- Основные векторы эксплуатации в ядре
- **DRAKVUF**
- Плагин exploitmon
- Обнаружение эксплойтов повышения привилегий
- Demo
- Заключение

DRAKVUF. Black-box Binary Analysis System

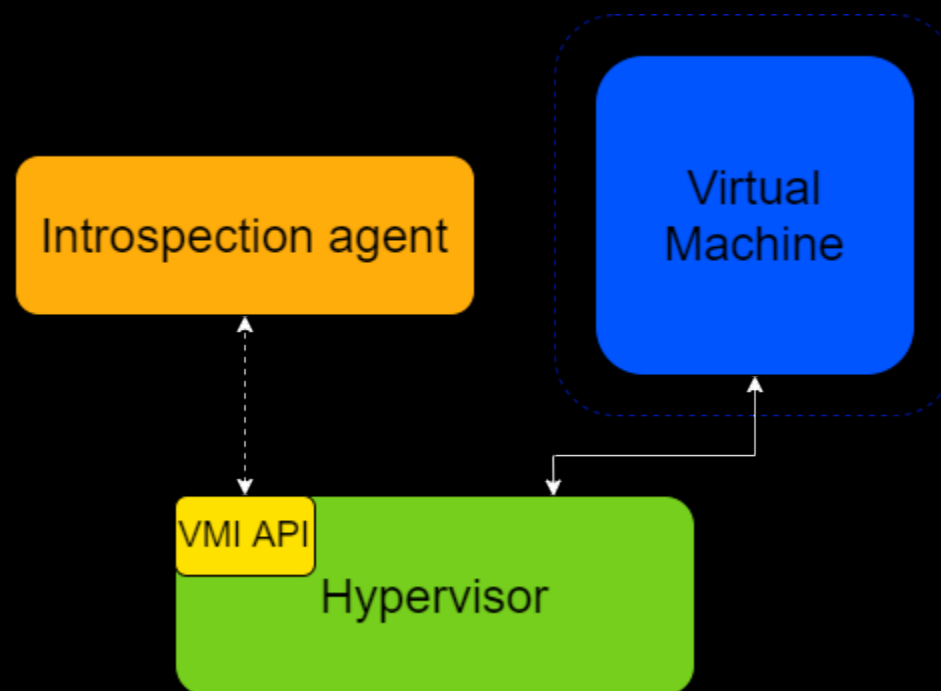


Возможность перехвата различных событий:

- Доступ к памяти (r/w/x)
- Прерывания
- Моделезависимые регистры (MSR)
- Регистры управления

Изменение состояния системы:

- Регистры VCPU
- Физическая память



DRAKVUF. Black-box Binary Analysis System



- Фреймворк для скрытного динамического анализа
- API не зависит от гостевой операционной системы
- Легко расширяется с помощью плагинов
- Использует LibVMi для мониторинга и изменения состояния виртуальной системы

LibVMI. Virtual Machine Introspection



- Модификация виртуализованной системы из гипервизора
- Перехват различных системных событий
- Обширный API на все случаи жизни

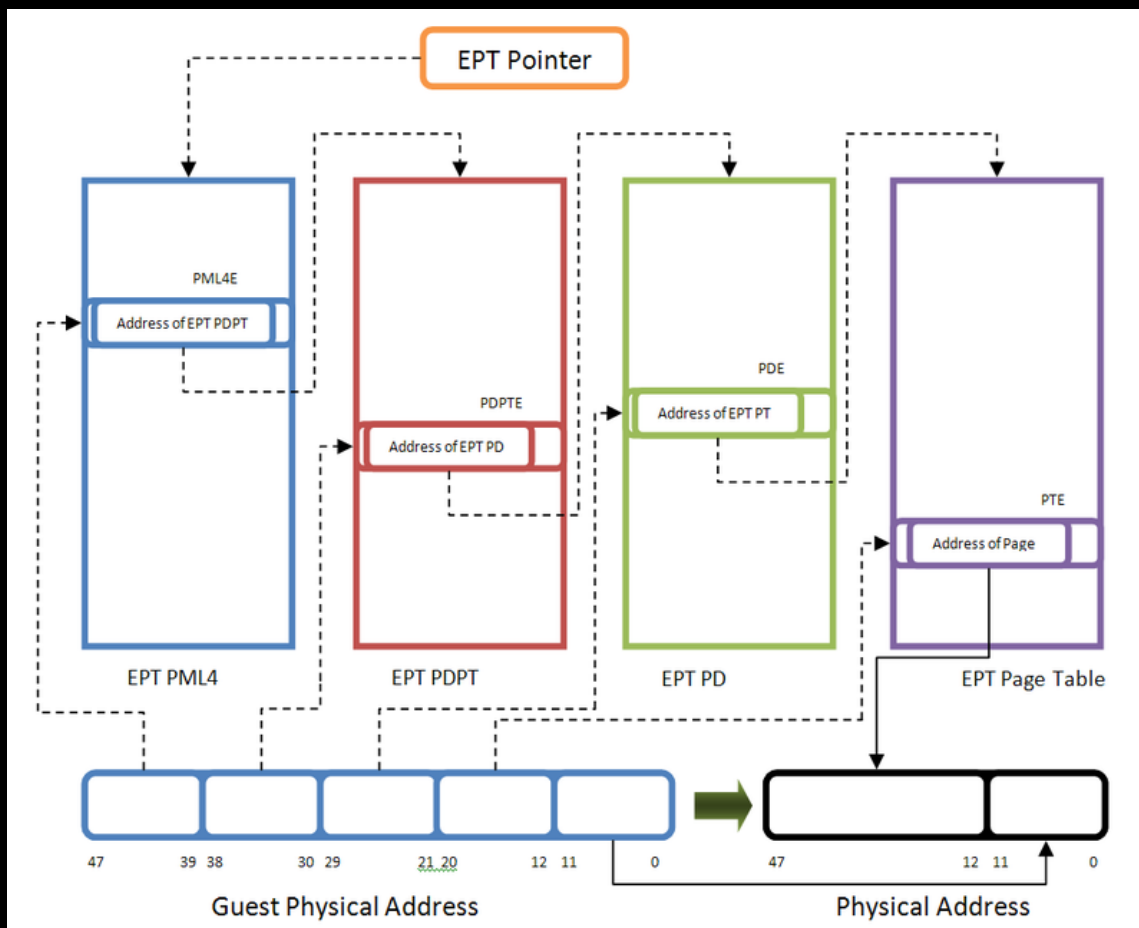
Functions

`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`

Перехваты EPT

PT

Трансляция памяти VM в память хоста



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

Перехваты EPT



Перехват событий

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

Трасса выполнения

```
#include <Windows.h>

unsigned char buffer[] = "\x90\x90\x90\x90";

int main()
{
    HANDLE hfile = CreateFile(L"suspicious_file", GENERIC_WRITE, 0, NULL, CREATE_NEW, 0, 0);
    WriteFile(hfile, buffer, sizeof(buffer), NULL, NULL);
    CloseHandle(hfile);
}
```

Пример программы

Трасса выполнения

ProcessName = users\\sample.exe, TID = 4004, PID = 3024, PPID = 1464,
FileName = users\\suspicious_file, plugin = filetracer, Method = NtCreateFile

ProcessName = users\\sample.exe, TID = 4004, PID = 3024, PPID = 1464,
FileName = users\\suspicious_file, plugin = filetracer, Method = NtWriteFile

Плагины



- memdump – сохранение потенциально интересных страниц виртуальной памяти.
- filetracer – отслеживание потенциально интересных операций с файлами.
- regmon – операции с реестром.
- syscalls – трассировка системных вызовов.
- apimon – перехват функций usermode: в частности, обращение к COM-интерфейсам и RPC.
-

Содержание



- Коротко о PT Sandbox
- Основные векторы эксплуатации в ядре
- DRAKVUF
- **Плагин exploitmon, обнаружение эксплойтов ядра**
- Demo
- Заключение

Exploitmon. Что это?



- Новый плагин для DRAKVUF
- Отслеживает изменения в динамических структурах ядра ОС
- Обнаруживает попытки эксплуатации ядра ОС

Обнаружение модификации HalDispatchTable

Как обнаруживать:

- Мониторинг записи на страницу памяти, где находится HalDispatchTable

```
349 static event_response_t hal_table_overwrite_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
350 {
351     addr_t hal_table_pa = reinterpret_cast<addr_t>(info->trap->data);
352     // https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/ntos/hal/hal_dispatch.htm
353     if (info->trap_pa >= hal_table_pa && info->trap_pa < hal_table_pa + 0x100)
354     {
355         fmt::print(plugin->format, "exploitmon", drakvuf, info,
356             keyval("Method", fmt::Qstr("HalDispatchTable overwrite detected")));
357     }
358     return VMI_EVENT_RESPONSE_NONE;
359 }
```

ProcessName = users\\sample.exe, TID = 4004, PID = 3024, PPID = 1464,
plugin = exploitmon, Method = HalDispatchTable overwrite detected

Обнаружение модификации токена безопасности

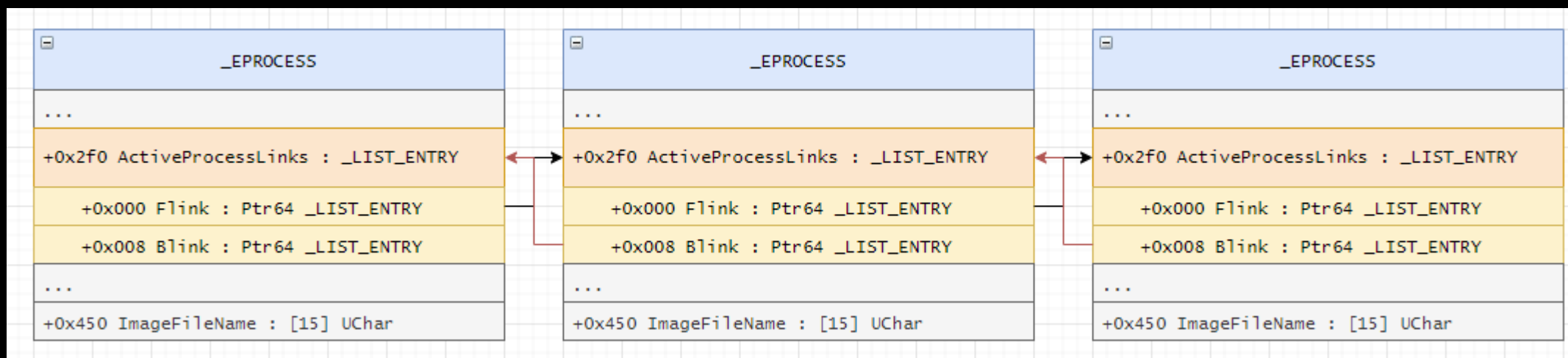


- Сохранять уже имеющиеся токены процессов на моменте запуска VM
- Сохранять токены на моменте создания процессов
- Сверять токен при завершении процесса или завершении анализа VM

Обнаружение модификации токена безопасности

- Сохранять уже имеющиеся токены процессов на моменте запуска VM

```
401 // Enumerate existing processes
402 drakvuf_enumerate_processes(drakvuf, process_visitor, static_cast<void*>(this));
```



<https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/manipulating-activeprocesslinks-to-unlink-processes-in-userland>

Обнаружение модификации токена безопасности

- Сохранять уже имеющиеся токены процессов на моменте запуска VM

```
352 static void process_visitor(drakvuf_t drakvuf, addr_t process, void* ctx)
353 {
354     auto plugin = static_cast<exploitmon*>(ctx);
355     // Read process token, pid
356     addr_t token, pid;
357     if (!read_kernel_addr(drakvuf, process + plugin->offsets[EPROCESS_TOKEN], &token))
358     {
359         PRINT_DEBUG("[EXPLOITMON] Failed to read process token\n");
360         throw -1;
361     }
362     if (!read_kernel_addr(drakvuf, process + plugin->offsets[EPROCESS_UNIQUE_PROCESS_ID], &pid))
363     {
364         PRINT_DEBUG("[EXPLOITMON] Failed to read process pid\n");
365         throw -1;
366     }
367     PRINT_DEBUG("[EXPLOITMON] Found process %lu\n", pid);
368     // Strip ref count bits
369     token = token & ~7;
370     plugin->processes.push_back(std::make_pair(pid, token));
371 }
```

Обнаружение модификации токена безопасности

- Сохранять токены на моменте создания процессов

int64 __fastcall PspInsertProcess (PEPROCESS Process,

xrefs to PsplInsertProcess

Direction	Type	Address	Text
Up	o	.rdata:000000001400450B0	RUNTIME_FUNCTION <rva
Up	o	.pdata:000000001400E8CCC	RUNTIME_FUNCTION <rva
Do...	p	NtCreateUserProcess+8FC	call PsplInsertProcess
Do...	p	PsCreateMinimalProcess+221	call PsplInsertProcess
Do...	p	PspCreateProcess+2A5	call PsplInsertProcess

Line 3 of 5

OK Cancel

```
250 static event_response_t insert_process_hook_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
251 {
252     addr_t process = drakvuf_get_function_argument(drakvuf, info, 1);
253     auto plugin = static_cast<exploitmon*>(info->trap->data);
254     // Read process token and pid
255     addr_t token, pid;
256     if (!read_kernel_addr(drakvuf, process + plugin->offsets[EPROCESS_TOKEN], &token))
257     {
258         PRINT_DEBUG("[EXPLOITMON] Failed to read process token\n");
259         throw -1;
260     }
261     if (!read_kernel_addr(drakvuf, process + plugin->offsets[EPROCESS_UNIQUE_PROCESS_ID], &pid))
262     {
263         PRINT_DEBUG("[EXPLOITMON] Failed to read process pid\n");
264         throw -1;
265     }
266     // Strip ref count bits
267     token = token & ~7;
268     plugin->processes.push_back(std::make_pair(pid, token));
269     return VMI_EVENT_RESPONSE_NONE;
270 }
```

Обнаружение модификации токена безопасности

- Сверять токен при завершении процесса или завершении анализа VM

int64 __fastcall PspTerminateProcess(PEPROCESS Process, __int64 a2, 237
xrefs to PspTerminateProcess 238

Direction	Type	Address	Text
Up	o	.rdata:00000000140046730	RUNTIME_FUNCTION <rva PspTerminateProces
Up	o	.pdata:000000001400E9158	RUNTIME_FUNCTION <rva PspTerminateProces
Up	p	NtTerminateProcess+97	call PspTerminateProcess 241
Do...	p	PsTerminateProcess+26	call PspTerminateProcess 242
Do...	p	PspTerminatePicoProcess+26	call PspTerminateProcess 243

Line 3 of 5 244 245 246 247 248 249 250 251 252 253

OK Cancel Search

```
237 for (const auto& [p_pid, p_token] : plugin->processes)
238 {
239     if (p_pid == pid)
240     {
241         if (p_token != token)
242         {
243             fmt::print(plugin->format, "exploitmon", drakvuf, info,
244                 keyval("Method", fmt::Qstr("Token modification detected")));
245         }
246         // Don't forget to remove it!
247         plugin->processes.erase(plugin->processes.begin() + i);
248         break;
249     }
250     i++;
251 }
252 return VMI_EVENT_RESPONSE_NONE;
253 }
```

Обнаружение модификации токена безопасности



ProcessName = users\\sample.exe, TID = 4004, PID = 3024, PPID = 1464,
plugin = exploitmon, Method = Token modification detected

Обнаружение ядерного потока в пользовательской памяти

- Легитимно ли исполнение ядерного потока в пользовательской памяти?

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of 4KB page frame																				Ignored		G	P A T	D	A	P C D	PW T	U / S	R / W	<u>1</u>	PTE: 4KB page	

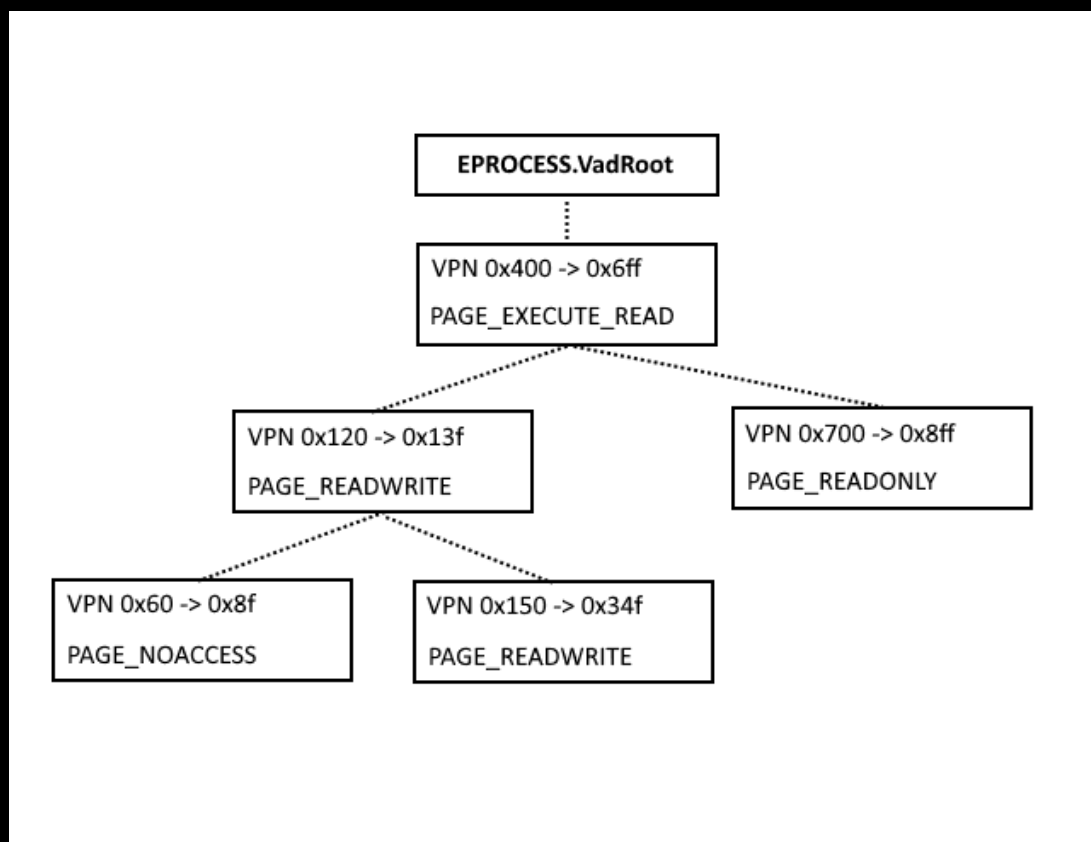
Return value

ExGetPreviousMode returns a KPROCESSOR_MODE value, one of `KernelMode` or `UserMode`. This value specifies the previous processor mode for the current thread.

Виртуальная память в Windows

- Процессы требуют память по необходимости
- Страница памяти 4КБ
- Выделение памяти всегда кратно 64КБ
- Виртуальная память не всегда имеет под собой физическую

Виртуальная память в Windows



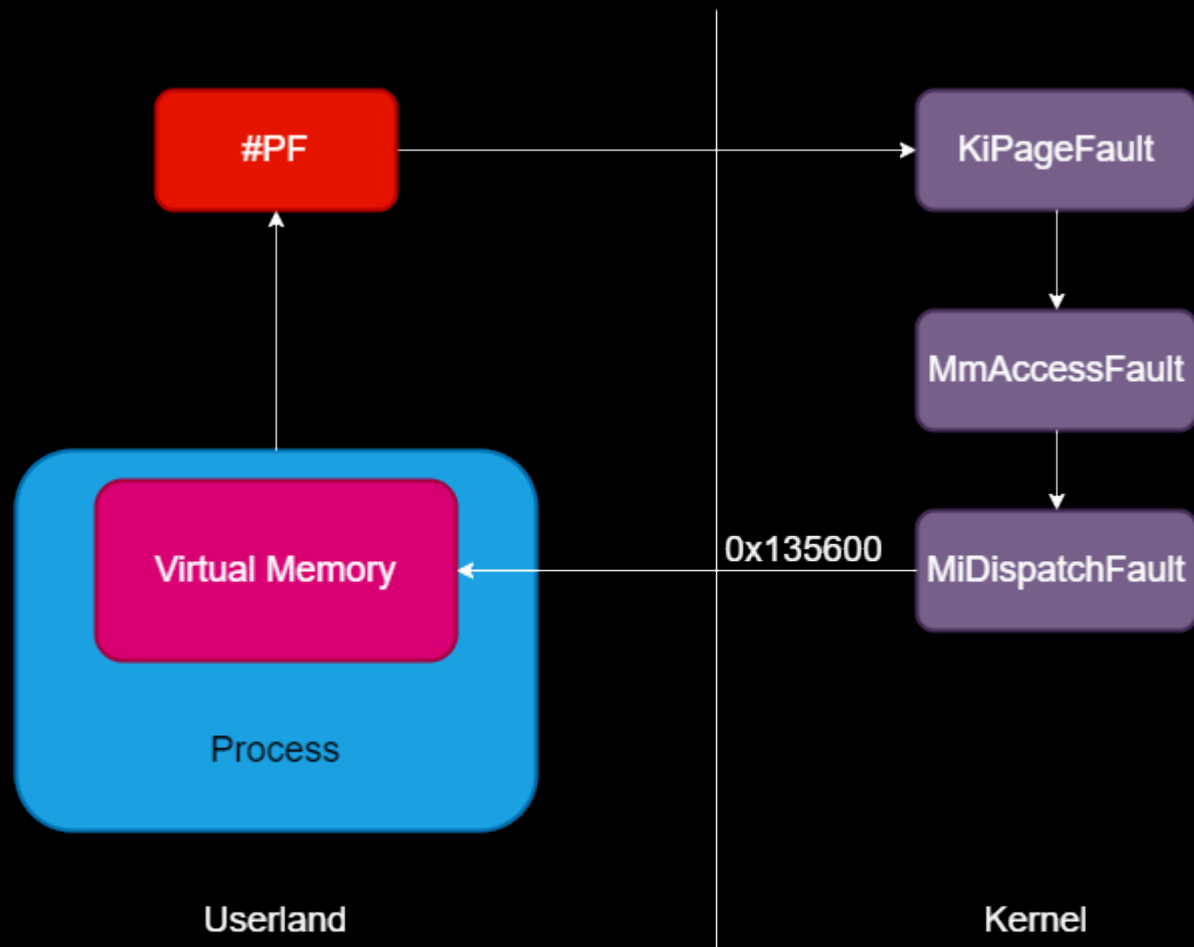
Виртуальная память в Windows

- Виртуальная память процесса описывается деревом VAD
- Указатель на корень дерева VAD находится в структуре EPROCESS
- Mapped VAD node – DLLs, mapped files, etc.
- Private VAD node – Private Heap

```
        UCHAR SubSystemMinorVersion;  
        UCHAR SubSystemMajorVersion;  
    };  
    WORD SubSystemVersion;  
};  
UCHAR PriorityClass;  
MM_AVL_TABLE VadRoot;  
ULONG Cookie;  
ALPC_PROCESS_CONTEXT AlpcContext;  
} EPROCESS, *PEPROCESS;
```

Жизнь page fault

PT



Обнаружение ядерного потока в пользовательской памяти

- Код должен быть быстрым
- Перехваты должны ставить только на страницы с правами на исполнение

Commit Charge (K)		Kernel Memory (K)		Paging Lists (K)	
Current	10 946 180	Paged WS	483 844	Zeroed	22 375 160
Limit	38 123 376	Paged Virtual	532 140	Free	8 916
Peak	12 798 012	Paged Limit	no symbols	Modified	359 436
		Nonpaged	947 404	ModifiedNoWrite	88
Peak/Limit	33.57%	Nonpaged Limit	16 777 216	Standby	1 203 012
Current/Limit	28.71%			Priority 0	8 484
				Priority 1	33 496
				Priority 2	30 944
				Priority 3	8 356
				Priority 4	122 300
				Priority 5	846 880
				Priority 6	0
				Priority 7	152 552
				PageFileModified	359 224
Physical Memory (K)		Paging			
Total	33 142 640	Page Fault Delta	3 364		
Available	23 587 088	Page Read Delta	1		
Cache WS	160 004	Paging File Write Delta	0		
Kernel WS	8	Mapped File Write Delta	0		
Driver WS	7 428				

Обнаружение ядерного потока в пользовательской памяти

- Перехватить MmAccessFault
- Получить физическую страницу
- Перехватить физическую страницу на исполнение
- PROFIT???

```
278 static event_response_t execute_faulted_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
279 {
280     drakvuf_remove_trap(drakvuf, info->trap, nullptr);
281     page_info_t p_info = {};
282     {
283         vmi_lock_guard lg(drakvuf);
284         if (VMI_SUCCESS != vmi_pagetable_lookup_extended(lg.vmi, info->regs->cr3, reinterpret
285             return VMI_EVENT_RESPONSE_NONE;
286     }
287     auto page_user = (p_info.x86_ia32e.pte_value) & 4;
288     privilege_mode_t mode;
289     if (!drakvuf_get_current_thread_previous_mode(drakvuf, info, &mode))
290         return VMI_EVENT_RESPONSE_NONE;
291
292     if (page_user ^ mode)
293     {
294         fmt::print(plugin->format, "exploitmon", drakvuf, info,
295             keyval("Method", fmt::Qstr("Kernel thread execution")));
```

Обнаружение ядерного потока в пользовательской памяти

PT

ProcessName = users\\sample.exe, TID = 4004, PID = 3024, PPID = 1464,
plugin = exploitmon, Method = Kernel thread execution

Содержание

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

- Коротко о PT Sandbox
- Техники манипуляций с динамическими объектами ядра
- DRAKVUF
- Плагин exploitmon
- Обнаружение эксплойтов повышения привилегий
- Demo
- Заключение

Полезные ссылки



PT Sandbox

ptsecurity.com/ru-ru/products/sandbox/



PT ESC Threat Intelligence blog

ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/



PT ESC Incident Response Alert

ptsecurity.com/ru-ru/services/esc/



Вопросы

webinar@ptsecurity.com

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/>

Закрепление и уклонение от обнаружения:
детектирование техник на примере PT Sandbox

ptsecurity.com/ru-ru/research/webinar/zakreplenie-i-uklonenie-ot-obnaruzheniya-detektirovanie-tehnik-na-primere-pt-sandbox/

Павел Максютин

pmaksyutin@ptsecurity.com

Алексей Вишняков

avishnyakov@ptsecurity.com

Twitter: @Vishnyak0v