501 – eBPF never forgets

Team Information

Team Name : HSPACE

Team Member : Jinung Lee, Beomjun Park, DoHyeon Kim, Soyoung Cho

Email Address : hspacedigitalforensicslab@gmail.com

Teams must:

- Provide a detailed, step-by-step description of their problem-solving approach to ensure reproducibility by another examiner.
- List all tools used to arrive at their conclusions.

Tools used:

Name:	VMware Workstation Pro	Publisher:	Broadcom
Version:	17.6.3		
URL:	https://www.vmware.com		

Name:	BandiZip	Publisher:	BandiSoft
Version:	7.40		
URL:	https://www.bandisoft.com/bandizip		

Name:	CMake	Publisher:	Kitware
Version:	4.1.1		
URL:	https://cmake.org/download/		

Name:	Visual Studio Community 2022	Publisher:	Microsoft Corporation
Version:	17.14.15		
URL:	https://visualstudio.microsoft.com/ko/		

Name:	Windows Driver Kit	Publisher:	Microsoft Corporation
Version:	10.0.26100		
URL:	https://learn.microsoft.com/ko-kr/windows-hardware/drivers/download-the-wdk		

Name:	ebpf-for-windows	Publisher:	Microsoft
Version:	0.18.0		
URL:	https://github.com/microsoft/ebpf-for-windows		

Name:	Extreme.Injector.	Publisher:	master131
Version:	3.7.3		
URL:	https://github.com/master131/ExtremeInjector		

문제 풀이

사용자가 악성 파일을 수신했을 당시 사용 중이던 브라우저의 이름과 버전은 무엇인가?

문제 파일에 동봉된 로그 파일에 따르면 사용자가 악성 파일을 다운로드 한 시점은 "2025-06-29T11:48:46.877185+00:00"으로 파악됩니다.

로그
{"source": "file", "operation": 6, "pid": 1372, "file_name": "₩₩Device₩₩HarddiskVolume3₩₩Users₩₩austinkim₩₩Downloads₩₩xmrig-6.24.0-windows-x64.zipd", "file_size": 2715848, "create_time": 133955075297845079, "last_access_time": 133955075304417137, "last_modified_time": 133955075304417137, "time": "2025-06-29T11:48:46.877185+00:00"}

해당 로그를 기점으로 이전 로그를 살펴보면 다음과 같이 Microsoft Edge 사용 기록을 확인할 수 있습니다.

로그
{"source": "file", "operation": 18, "pid": 8572, "file_name": "₩₩Device₩₩HarddiskVolume3₩₩Program Files (x86)₩₩Microsoft₩₩Edge₩₩Application₩₩137.0.3296.93₩₩msedge_elf.dll", "file_size": 3802184, "create_time": 133955005106121801, "last_access_time": 133955075300802525, "last_modified_time": 133948393366868038, "time": "2025-06-29T11:48:46.870086+00:00"}

따라서, 사용자는 137.0.3296.93 버전의 Microsoft Edge 브라우저를 사용중인 것으로 파악됩니다.

정답	
브라우저 이름	Microsoft Edge
브라우저 버전	137.0.3296.93

악성 파일이 통신을 시도한 IP주소와 포트 번호는 무엇인가?

모네로 코인 채굴 악성 파일이 동작함과 동시에 모네로 마이닝풀 포트인 3333과 연결되는 로그를 확인할 수 있었습니다.

로그
{"source": "file", "operation": 3, "pid": 4024, "file_name": "₩₩Device₩₩HarddiskVolume3₩₩Users₩₩austinkim₩₩Desktop₩₩xmrig-6.24.0₩₩xmrig.exe", "file_size": 6455296, "create_time": 133951637460000000, "last_access_time": 133955075461703580, "last_modified_time": 133955075361444486, "time": "2025-06-29T11:48:47.446102+00:00"}
{"source": "network", "src_ip": "172.21.167.113", "src_port": 50148, "dst_ip": "178.128.242.134", "dst_port": 3333, "protocol": "TCP", "time": "2025-06-29T11:48:47.446107+00:00"}

따라서, 악성 파일이 통신을 시도한 IP와 포트는 다음과 같습니다.

정답	
IP	178.128.242.134
PORT	3333

악성 파일의 파일 이름, PID, PPID, 생성 시각은 무엇인가?

악성 파일인 xmrig.exe 프로세스가 시작된 시간은 다음 로그를 통해 확인할 수 있습니다.

로그
{"source": "process", "operation": 0, "pid": 1244, "image_file_name": "₩₩??₩₩C:₩₩Users₩₩austinkim₩₩Desktop₩₩xmrig-6.24.0₩₩xmrig.exe", "command_line": "xmrig.exe", "ppid": 7236, "created_at": "06/27/2025 07:19:06", "time": "2025-06-29T11:48:47.429690+00:00"}

따라서, 악성 파일 이름, PID, PPID, 생성 시각은 다음과 같습니다.

정답	
파일 이름	xmrig.exe
PID	1244
PPID	7236
생성 시각	2025-06-27 07:19:06 (UTC+9)

악성 파일을 실행한 프로그램의 이름, PID, PPID, 생성 시각은 무엇인가?

악성 파일인 xmrig.exe 프로세스의 PPID는 7236이고, xmrig.exe가 실행되기 직전에 PID가 7236인 cmd.exe가 실행되었습니다.

로그
{"source": "process", "operation": 0, "pid": 7236, "image_file_name": "₩₩??₩₩C:₩₩Windows₩₩System32₩₩cmd.exe", "command_line": "₩"C:₩₩Windows₩₩System32₩₩cmd.exe₩"", "ppid": 7132, "created_at": "06/27/2025 07:19:04", "time": "2025-06-29T11:48:47.421857+00:00"}
{"source": "process", "operation": 0, "pid": 1244, "image_file_name": "₩₩??₩₩C:₩₩Users₩₩austinkim₩₩Desktop₩₩xmrig-6.24.0₩₩xmrig.exe", "command_line": "xmrig.exe", "ppid": 7236, "created_at": "06/27/2025 07:19:06", "time": "2025-06-29T11:48:47.429690+00:00"}

따라서, 악성 파일을 실행한 프로그램의 이름, PID, PPID, 생성 시각은 다음과 같습니다.

정답	
파일 이름	cmd.exe
PID	7236
PPID	7132
생성 시각	2025-06-27 07:19:04 (UTC+9)

악성코드가 참조한 설정 파일의 이름과 마지막 수정 시각은 무엇인가?

악성 파일인 xmrig.exe은 채굴에 필요한 설정 값을 같은 경로의 config.json파일에 저장합니다. config.json 파일의 마지막 로그는 다음과 같습니다.

로그
{"source": "file", "operation": 6, "pid": 4, "file_name": "₩₩Device₩₩HarddiskVolume3₩₩Users₩₩austinkim₩₩Desktop₩₩xmrig-6.24.0₩₩config.json", "file_size": 3994, "create_time": 133951636480000000, "last_access_time": 133955075464971609, "last_modified_time": 133955075461843650, "time": "2025-06-29T11:48:47.487636+00:00"}

따라서, 악성코드가 참조한 설정 파일의 이름과 마지막 수정 시각은 다음과 같습니다.

정답	
파일 이름	config.json
마지막 수정 시각	2025-06-27 23:19:06.1843650 (UTC+9)

Windows 환경에서 DLL 인젝션을 탐지하기 위한 적절한 eBPF 기반 후킹 포인트를 선정하고, 그 선택 근거를 제시하라

(1) PsSetLoadImageNotifyRoutine

PsSetLoadImageNotifyRoutine은 프로세스 주소공간에 매핑되는 모든 PE 파일에 대해 커널이 등록된 콜백을 호출하여 즉시 해당 사실을 알립니다. 이를 통해 만약 DLL Injection이 일어난다면, 새 모듈이 로드 된 지점에서 이미지 경로, PID, 사이즈, 커널 혹은 유저 여부 등을 받아 콜백으로 받아 즉시 필터링을 수행할 수 있습니다.

예를 들어, "Temp"와 같은 임시 디렉터리나 네트워크 경로 등 의심되는 경로에서 로드되는 모듈, 서명 부재, 단시간 내 다량의 DLL로드 등의 징후를 포착하여 탐지할 수 있을 것으로 판단됩니다. 하지만 이는 Reflective Dll Injection과 같이 메모리에서 직접 매핑되는 경우 본 방법으로 탐지할 수 없을 것입니다.

(2) PsSetCreateThreadNotifyRoutine

PsSetCreateThreadNotifyRoutine은 DLL Injection 체인의 원격 스레드 생성 (CreateRemoteThread/NtCreateThreadEx)을 탐지할 수 있습니다.

스레드 생성 이벤트 수신 후 NtQueryInformationThread 및 VirtualQueryEx 등을 통해 시작 주소와 메모리 유형 및 보호 속성을 확인합니다. 이를 모듈 맵과 대조해 모듈 외부 시작 또는 MEM_PRIVATE의 실행 가능 구역이면 DLL Injection이 일어난 것으로 의심할 수 있습니다. 또한 생성자 PID와 대상 PID가 다른 원격 스레드이면서, 동일 타임라인에 권한 상승 및 메모리 조작 이벤트가 동반된다면, 이는 DLL Injection이 일어난 것으로 의심할 수 있습니다.

(3) ObRegisterCallbacks

ObRegisterCallbacks를 통한 프로세스 및 스레드 핸들 감시는 Injection 체인을 확인할 수 있습니다. Injection이 일어날 때는 보통 대상 프로세스에 대해 PROCESS_VM_WRITE, PROCESS_CREATE_THREAD, PROCESS_SUSPEND_RESUME 등의 권한을 요청합니다.

커널 오브젝트 매니저 콜백에서 이러한 권한 비트를 실시간으로 관찰하면, "권한 취득 → 메모리 조작 → 원격 스레드"로 이어지는 체인을 재구성할 수 있습니다. 동일 PID가 여러 대상 PID에 높은 권한의 핸들을 연쇄적으로 시도하는 패턴은 DLL Injection 시도가 일어난 것으로 의심할 수 있습니다.

따라서 Windows 환경에서 DLL 인젝션을 탐지하기 위한 적절한 eBPF 기반 후킹 포인트는 위 3

개로 선정할 수 있습니다.

정답	
후킹 포인트	PsSetLoadImageNotifyRoutine
	PsSetCreateThreadNotifyRoutine
	ObRegisterCallbacks

6번에서 제시한 전략을 바탕으로 실제로 동작하는 eBPF 확장 모듈(extension)을 구현하라 ※ 구현한 프로젝트 파일을 별첨으로 첨부했으니 참고 부탁드립니다.

문제를 풀기 위한 가상머신 구축에 사용된, 윈도우의 정보는 아래의 표와 같습니다.

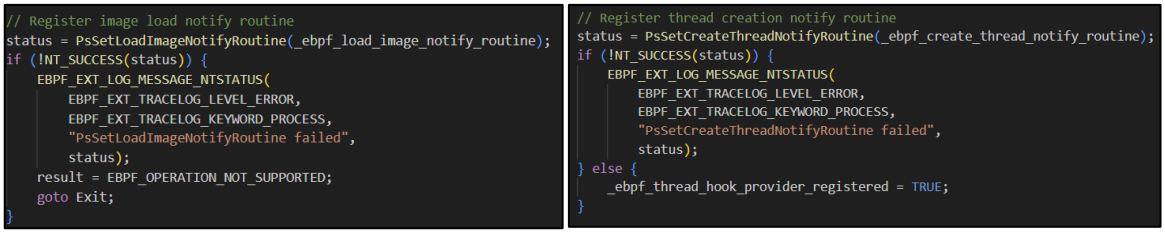
구분	정보
Windows	Windows 11
버전	24H2
OS 빌드	26100.6584

관련 내용을 찾던 중, capelabs의 eBPF-for-DFIR 깃허브를 발견할 수 있었습니다. 이는 Windows 시스템에서 디지털 포렌식 및 사고 대응(DFIR)을 위한 실시간 시스템 데이터를 수집하는 오픈소스 도구이며, eBPF-for-Windows 프레임워크를 활용하여 시스템 활동에 대한 심층적인 가시성을 제공하고 사고 대응 담당자가 조사 과정에서 중요한 정보를 수집할 수 있도록 지원합니다.

원래 해당 도구의 목적은, 아래 그림과 같이 PsSetCreateProcessNotifyRoutineEx를 이용하여 프로세스의 생성과 종료를 모니터링하는 도구입니다.

```
if (!_ebpf_process_hook_provider_registered) {
    // Register the process create notify routine.
    NTSTATUS status = PsSetCreateProcessNotifyRoutineEx(_ebpf_process_create_process_notify_routine_ex, FALSE);
    if (!NT_SUCCESS(status)) {
        EBPF_EXT_LOG_MESSAGE_NTSTATUS(
            EBPF_EXT_TRACELOG_LEVEL_ERROR,
            EBPF_EXT_TRACELOG_KEYWORD_PROCESS,
            "PsSetCreateProcessNotifyRoutineEx failed",
            status);
        result = EBPF_OPERATION_NOT_SUPPORTED;
        goto Exit;
    }
    _ebpf_process_hook_provider_registered = TRUE;
}
```

하지만 DLL Injection 탐지를 위해 기존의 프로세스 모니터링에서 PsSetLoadImageNotifyRoutine 과 PsSetCreateThreadNotifyRoutine 을 이용하여 DLL 로드 이벤트와 스레드 생성 이벤트까지 확장 모니터링하도록 변경했습니다. 이러한 변경을 통해 CreateRemoteThread 기반의 DLL Injection 공격에서 발생하는 핵심 이벤트들을 실시간으로 탐지할 수 있게 되었습니다.



```
// Register image load notify routine
status = PsSetLoadImageNotifyRoutine(_ebpf_load_image_notify_routine);
if (!NT_SUCCESS(status)) {
    EBPF_EXT_LOG_MESSAGE_NTSTATUS(
        EBPF_EXT_TRACELOG_LEVEL_ERROR,
        EBPF_EXT_TRACELOG_KEYWORD_PROCESS,
        "PsSetLoadImageNotifyRoutine failed",
        status);
    result = EBPF_OPERATION_NOT_SUPPORTED;
    goto Exit;
}
```

```
// Register thread creation notify routine
status = PsSetCreateThreadNotifyRoutine(_ebpf_create_thread_notify_routine);
if (!NT_SUCCESS(status)) {
    EBPF_EXT_LOG_MESSAGE_NTSTATUS(
        EBPF_EXT_TRACELOG_LEVEL_ERROR,
        EBPF_EXT_TRACELOG_KEYWORD_PROCESS,
        "PsSetCreateThreadNotifyRoutine failed",
        status);
} else {
    _ebpf_thread_hook_provider_registered = TRUE;
}
```

또한 오탐과 악성코드와 유사한 패턴을 탐지하기 위해서도 노력했습니다.

오탐의 경우 아래 표와 같이 3가지의 주요 필터링을 통해 오탐을 줄이기 위해 노력했습니다.

기법	코드

시스템 이미지 제외	if (ImageInfo->SystemModeImage) {
	return; // Skip kernel images
	}

시스템 프로세스 제외	if (IsSystemProcess(ProcessId) || IsSystemProcess(current_pid)) {
	return;
	}

시스템 DLL 제외	static BOOLEAN IsSystemDll(PUNICODE_STRING ImagePath)
	{
	if (!ImagePath || !ImagePath->Buffer) return FALSE;
	// Check for Windows system directories
	if (wcsstr(ImagePath->Buffer, L"₩₩Windows₩₩System32₩₩") ||
	wcsstr(ImagePath->Buffer, L"₩₩Windows₩₩SysWOW64₩₩") ||
	wcsstr(ImagePath->Buffer, L"₩₩Windows₩₩Microsoft.NET₩₩") ||
	wcsstr(ImagePath->Buffer, L"₩₩Windows₩₩WinSxS₩₩")) {
	return TRUE;
	}

	return FALSE; }

또한 6번과 같이 악성코드와 유사한 패턴을 감지하기 위해서 2가지의 필터링 로직을 통하여 탐지율을 높였습니다.

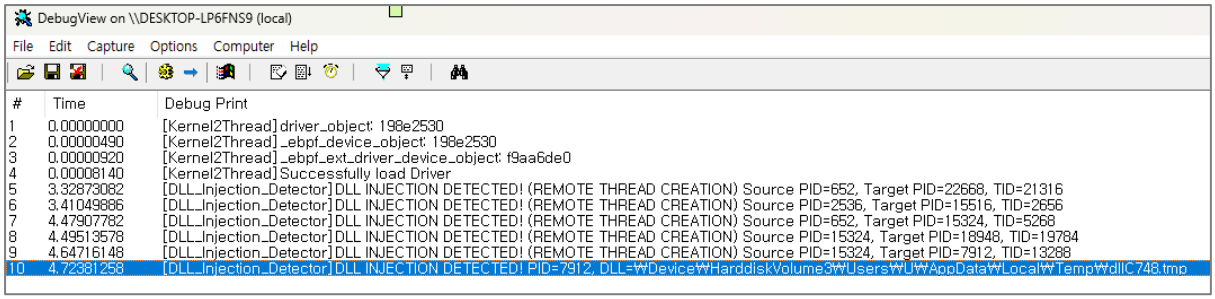
기법	코드

임시 파일 탐지	if (FullImageName != NULL && FullImageName->Buffer != NULL) {
	if (wcsstr(FullImageName->Buffer, L".tmp") ||
	wcsstr(FullImageName->Buffer, L"₩₩Temp₩₩")) {
	is_suspicious = TRUE;
	}
	}

원격 프로세스 체크	if (injection_context.process_md.is_remote) {
	// 최근 5초 내 핸들 오픈 이력 확인
	if ((current_time.QuadPart - tracker->timestamp.QuadPart) < (5LL * 10000000LL)) {
	is_suspicious = TRUE;
	}
	}

제공된 멀웨어(injector.exe)를 실행하여, 7번에서 구현한 extension이 DLL Injection을 올바르게 탐지하는지 증명하라

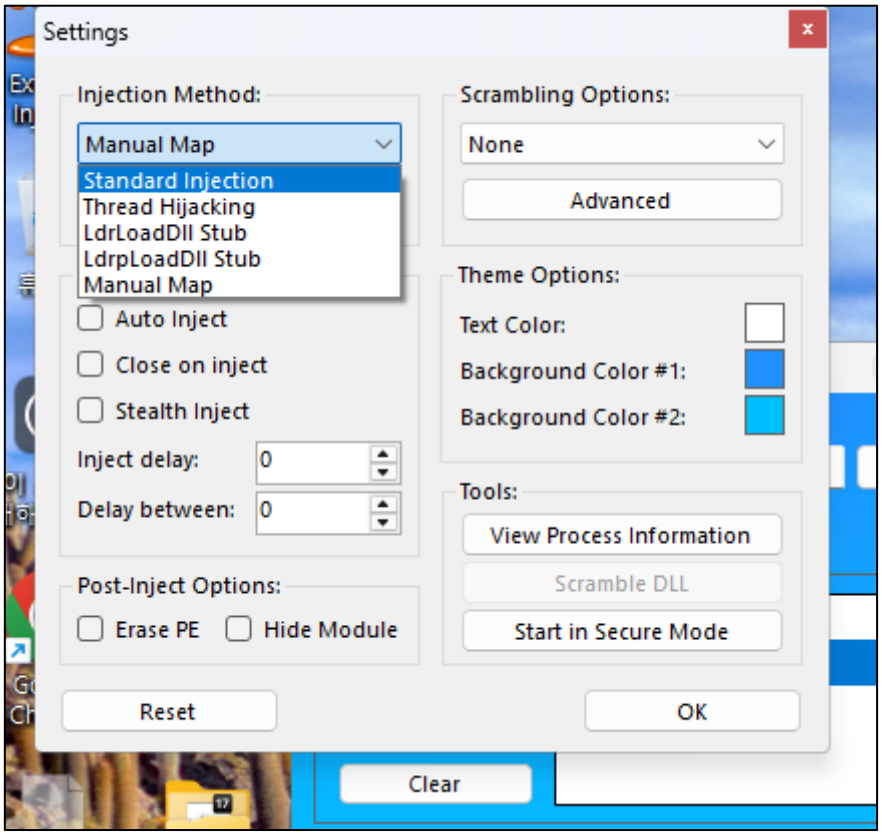
구현한 extension을 테스트한 결과, Injector.exe를 탐지하는 것으로 확인되었습니다. 탐지 과정을 녹화한 영상은 별첨으로 첨부했으니 참고 부탁드립니다.



구현한 extension 이 제공된 injector.exe 외에도 다양한 DLL injection 기법을 정상적으로 탐지할 수 있는지 검증하라. (100 점)

인젝션 도구인 Extreme.Injector 를 이용하여 탐지를 시도했습니다.

탐지에 성공한 기법은 Standard Injection, Thread Hijacking, LdrLoadDll Stub, Manual Map 으로 총 4 개의 기법 탐지에 성공했으며, 탐지 과정을 녹화한 영상은 별첨으로 첨부했으니 참고 부탁드립니다.



별첨 영상 및 소스코드

https://naver.me/F9hOfMrc

비밀번호: ebpf501