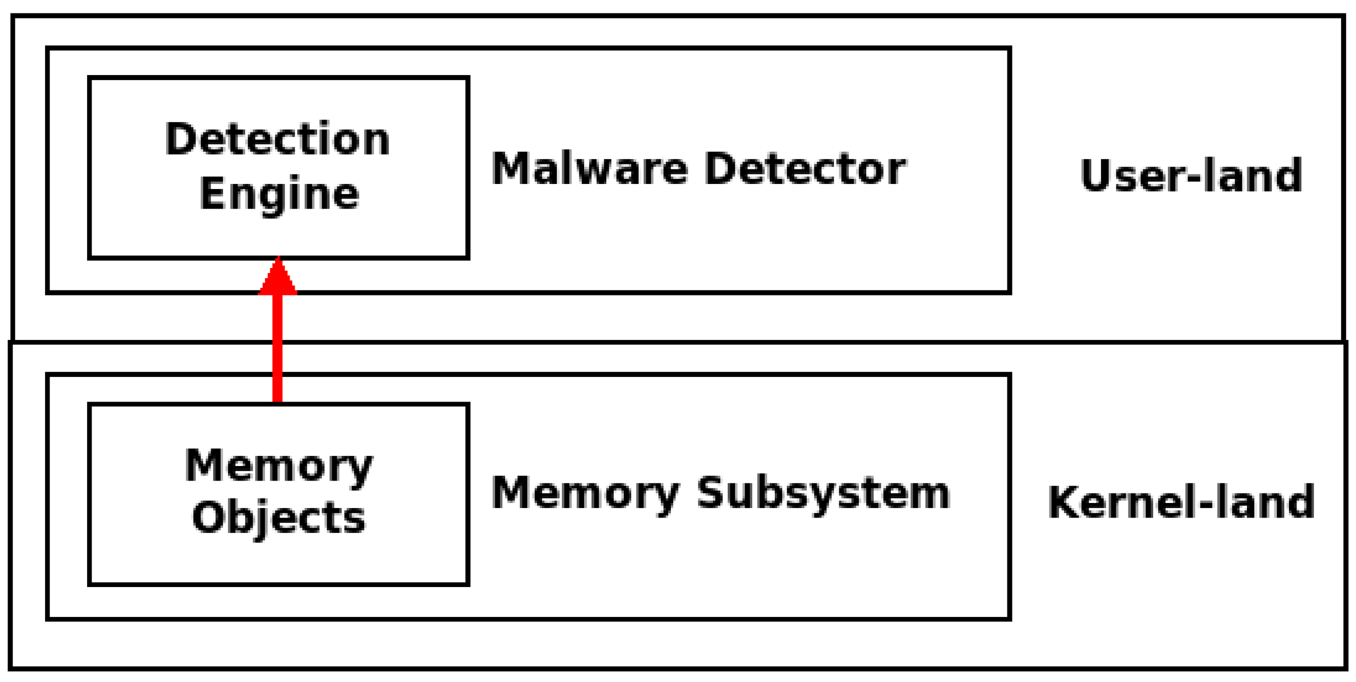
Detecting Memory Injections Using a Hardware Monitor

Marcus Botacin[1,2], Uriel Kosayev[2], and Amichai Yifrach[2]
[1]Texas A&M University (TAMU), USA and [2]Cymdall, Israel
[1]botacin@tamu.edu and [2]{uriel,amichai}@cymdall.com

Abstract

Memory injection is the current State-Of-The-Art (SOTA) malware attack technique. Injections are hard to detect by current software-based AntiViruses (AVs) because monitoring operations system-wide causes significant performance impact. To mitigate performance penalties, AntiViruses often only monitor specific parts of the system, thus naturally missing some injection points, that are actively exploited by the attackers in an arms race. A solution to the problem is to move AntiViruses to hardware to allow full-system monitoring without performance impact. We here present a prototype of a hardware monitor to detect memory injection attacks. We evaluate the prototype via the injection of a backdoor payload into a native Windows process. The injection is not detected by the native Windows Defender nor by commercial Endpoint Detection and Response (EDR) solutions, but it is detected by the proposed detector.

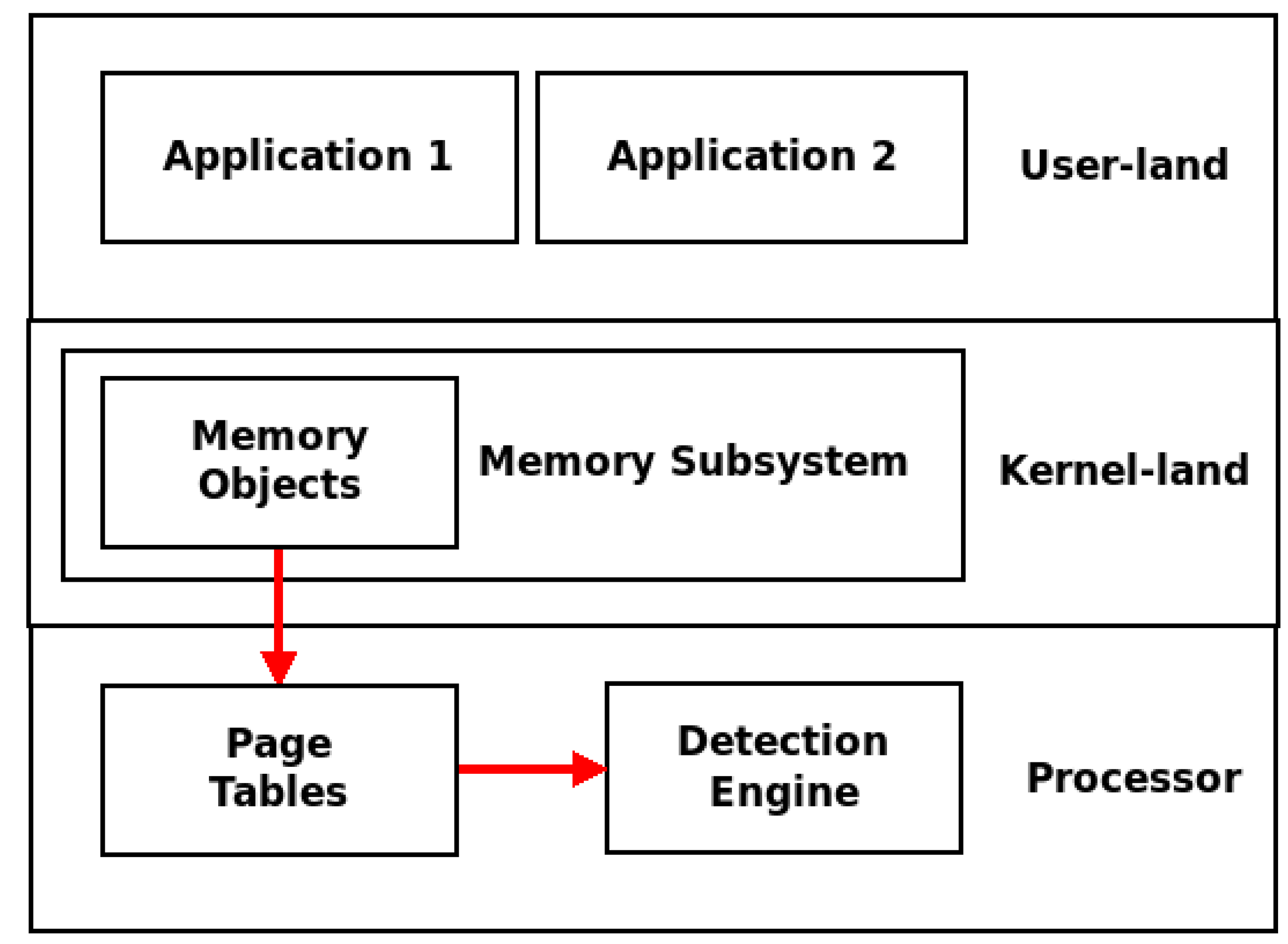
Software Detectors



Detection Results

Solution	Hardware	Defender	EDR1	EDR2
Detection	✓	✗	✗	✗

Hardware Detectors



Memory Injection Attacks

Most recent attacks are memory based because:

- Multi-stage malware is commonplace.
- Fileless malware is SOTA.

Memory injection attacks pose detection challenges, because:

- Processes may act maliciously any time.
- No file in disk for preliminary inspection.

A smart detection approach involves:

- Continuously monitoring memoery.
- Mapping data changes into intents.

Hardware vs. Software Detectors

Software detectors have drawbacks:

- Monitor causes performance impact.
- The monitor is a vulnerable surface.

Hardware detectors impose an implementation challenge:

- Semantic Gap: Software data structures must be reconstructed in hardware.

Hardware detectors have multiple advantages:

- Monitor causes no performance impact.
- Monitor has no internal attack surface.

Monitored Intents

```
enum intent_event_type_e : size_t
{
    INTENT_EVENTS
    // "New VAD created"
    #define X(new_vad)

    // "New VAD created without FileObject"
    #define X(new_vad_no_fileobject)

    // "Executable VAD became Write Executable"
    #define X(executable_vad_became_write_executable)

    // "Section added to memory PE"
    #define X(pe_section_added)
};
```

Kernel Data Structures Reconstruction

```
typedef union _EPROCESS_x64_10_19041_508_u{
    struct _EPROCESS_x64_10_19041_508
    {
        struct _KPROCESS_x64_10_19041_508 Pcb;
        struct _EX_PUSH_LOCK_x64_10_19041_508 ProcessLock;
        ...
        ULONG Flags;
            struct
            {
                ULONG CreateReported : 1;
                ULONG NoDebugInherit : 1;
                ULONG ProcessExiting : 1;
                ULONG ProcessDelete : 1;
                ULONG ManageExecutableMemoryWrites : 1;
```

Solution Console

Time	Message
00:03:48.115	Process Created. PID=4868; PPID=588; CPID=0; cmdLine: explorer.exe
06:26:59.230	Maliciouse Intent Probability 75.0 due to: Executable VAD FileObject changed in VAD node at 0x0000000000003170
06:26:59.230	Maliciouse Intent Probability 1.0 due to: VAD_SHORT changed to VAD in VAD node at 0x0000000000003170
06:26:59.230	Maliciouse Intent Probability 75.0 due to: Executable VAD FilePath changed in VAD node at 0x0000000000003170
06:26:59.230	Maliciouse Intent Probability 75.0 due to: New Write Executable VAD_SHORT created without FileObject - Injection found in VAD node at 0x00000000000031C0
06:26:59.230	Maliciouse Intent Probability 1.0 due to: VAD changed to VAD_SHORT in VAD node at 0x000000000000D540
06:26:59.230	Maliciouse Intent Probability 75.0 due to: Non Executable VAD_SHORT became Write Executable in VAD node at 0x000000000000D540
06:26:59.230	Maliciouse Intent Probability 75.0 due to: New Write Executable VAD_SHORT created without FileObject - Injection found in VAD node at 0x000000000000D540

Future Work

- **FPGA Prototyping.**
  - Parse the Windows kernel data structures in the hardware.
- **ASIC Prototyping.**
  - Convert the FPGA prototype into an energy-space efficient chip.
- **PCI Accelerator.**
  - Distribute the ASIC as a PCI board, security accelerator for easy integration.
- **Cloud Deployment.**
  - Deploy the solution at scale in partner cloud service providers.

References

1 Marcus Botacin et al. 2022. Terminator: A Secure Coprocessor to Accelerate Real-Time AntiViruses Using Inspection Breakpoints, ACM TOPS.

2 Marcus Botacin et al. 2022. HEAVEN: A Hardware-Enhanced AntiVirus ENgine to accelerate real-time, signature-based malware detection. Expert Systems with Applications.

3 Ashkan Hosseini. 2017. Ten process injection techniques: A technical survey of common and trending process injection techniques. Endpoint Security Blog (2017).

4 Metasploit. 2020. How to use a reverse shell in Metasploit. https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-a-reverse-shell-in-metasploit.html.

5 MITRE. 2020. Process Injection: Asynchronous Procedure Call.