

# **Lecture 6 Notes- Network and System Security**

## **1. What Malware Analysts Do**

### **Core Responsibilities**

- Investigate suspicious files and behaviors to assess capability, intent, and impact
- Analyse using three approaches: static, dynamic, and code analysis
- Produce actionable outputs for defenders and incident responders

### **Key Outputs**

IOCs (Indicators of Compromise):

- File hashes
- Network indicators (domains, IPs)
- System artifacts (mutexes, registry paths, certificates)
- DGA (Domain Generation Algorithm) seeds

Detections:

- YARA rules for file/memory scanning
- Sigma rules for SIEM log analysis
- MITRE ATT&CK framework mapping

Impact: Faster incident triage, targeted blocking, clearer IR playbooks, reduced organizational risk

## **2. End-to-End Analysis Workflow**

### **Stage 1: Intake & Static Triage**

- Receive sample hash, family hints, priority level
- Examine PE/ELF/Mach-O headers and sections
- Extract imports, strings, and entropy metrics
- Identify packers using tools like DIE
- Generate capability hints with capa tool

### **Stage 2: Dynamic Analysis**

- Execute in isolated environment (VM with snapshots)
- Monitor with ProcMon/Sysmon for:
  - Filesystem changes
  - Registry modifications
  - Process behavior
- Capture network traffic with Wireshark (PCAP analysis)

### **Stage 3: Unpacking & Patching**

- Defeat packing and obfuscation layers
- Bypass anti-debugging techniques
- Overcome anti-VM detection

### **Stage 4: Code Reverse Engineering**

- Recover control flow and data flow
- Identify key handlers, crypto routines, and triggers
- Understand malware logic and decision points

### **Stage 5: Config & IOC Extraction**

- Extract C2 (Command & Control) infrastructure
- Recover encryption keys
- Identify DGA seeds
- Document persistence mechanisms
- Collect forensic artifacts

### **Stage 6: Detection & Reporting**

- Write YARA/Sigma detection rules
- Map behaviors to ATT&CK framework
- Document remediation steps
- Create reports for technical and non-technical audiences

## **3. Core Competencies**

### **Part I: Foundational Skills**

#### **Behavioral Analysis:**

- Lab hygiene: VM isolation, regular snapshots
- Process monitoring with ProcMon and Sysinternals suite
- Telemetry collection using Sysmon

#### **Static Triage:**

- Binary format analysis (PE, ELF, Mach-O)
- Import/export table examination
- String extraction and analysis
- Entropy calculation and packer identification
- YARA-based triage

#### **Code Reverse Engineering:**

- x86/x64 assembly language
- .NET Intermediate Language (IL)
- Disassemblers: IDA Pro, Ghidra
- Debuggers: x64dbg, WinDbg
- Control Flow Graph (CFG) analysis
- Data flow reasoning

### **De-obfuscation & Evasion Defeat:**

- CFG deflattening
- Multi-layer unpacking
- API hash resolution
- Anti-debug bypass
- Anti-VM circumvention

### **Part II: Advanced Capabilities**

#### **Memory & Config Extraction:**

- Memory forensics with Volatility/Rekall
- Process dumping techniques
- Carving keys, URLs, and data structures from RAM

#### **Document & Script Malware:**

- Office macros (VBA)
- PDF JavaScript
- LNK file analysis
- PowerShell script examination
- Safe sandboxing practices

#### **Network/C2 Analysis:**

- Beacon timing patterns
- Protocol and TLS fingerprinting
- URI pattern identification
- PCAP-to-IOC extraction pipeline

#### **Reporting & Detection Engineering:**

- Clear technical writing for diverse audiences
- Actionable, engineerable detection rules
- MITRE ATT&CK technique mapping

## **4. Essential Toolchain**

- Disassemblers/Decompilers: IDA Pro, Ghidra
- Debuggers: x64dbg, WinDbg
- System Monitoring: Sysinternals Suite (ProcMon, Autoruns)
- Network Analysis: Wireshark
- Memory Forensics: Volatility
- Capability Detection: capa
- Pattern Matching: YARA
- .NET Analysis: dnSpy, dnlib
- Scripting/Automation: Python

## **5. Breaking Into Malware Analysis**

### **Lab Setup**

- Build isolated analysis environment
- Critical: No bridged network adapters by default
- Maintain meticulous notes
- Take regular VM snapshots before each analysis

### **Foundation Knowledge**

1. Operating System Internals (especially Windows)
2. Assembly Language (x86/x64) and IL
3. Binary Formats (PE structure, .NET metadata)
4. Windows API understanding
5. Networking Fundamentals