Lecture 6 – Reverse Engineering in Politics

Networks and System Security

Guest Lecture by Bamidele Ajayi

About the Speaker

- Based in Canada
- PhD Researcher (AI/ML Security), University of Sunderland
- Focus: malware analysis, reverse engineering, AI-assisted detection
- Research: Artificial Intelligence for Malware Detection

What Malware Analysts Do

Core Responsibilities

1. **Investigate suspicious files/behaviors**
   - Assess capability, intent, and impact
   - Build evidence and confidence through analysis
2. **Multi-layered analysis approach**
   - Static analysis
   - Dynamic analysis
   - Code analysis
3. **Produce actionable outputs**
   - IOCs (Indicators of Compromise)
   - Detections (YARA/Sigma rules)
   - Remediation guidance for defenders

Key Outputs & Impact

IOCs (Indicators of Compromise)

- File hashes
- Domains/IP addresses
- Mutexes
- Registry paths
- Certificates
- DGA (Domain Generation Algorithm) seeds

Goldsmiths
UNIVERSITY OF LONDON

Detections

- YARA rules: for files/memory analysis
- Sigma rules: for SIEM logs
- ATT&CK mapping: categorize attack techniques

Impact on Security

- Faster triage of threats
- Targeted blocking capabilities
- Clearer incident response (IR) playbooks
- Overall risk reduction

End-to-End Workflow

1. Intake

- Receive file hash
- Family hints (known malware families)
- Quick triage and priority assessment

2. Static Triage

- **Headers/sections**: analyze file structure
- **Imports**: identify imported functions
- **Strings**: extract readable text
- **Packer ID**: identify if file is packed/compressed
- **Capability hints**: use tools like `capa` to detect capabilities

3. Dynamic Run (Isolated Environment)

- **ProcMon/Sysmon**: monitor process activity
- **Filesystem/registry changes**: track modifications
- **PCAP (Wireshark)**: capture network traffic

4. Unpack/Patch

- Defeat packing/obfuscation techniques
- Bypass anti-debug checks

- Bypass anti-VM (virtual machine) checks

5. Code Reverse Engineering

- Recover control flow and data flow
- Find handlers, cryptographic functions, triggers
- Understand malware behavior at code level

6. Config & IOC Extraction

- C2 (Command & Control) servers
- Encryption keys
- DGA seeds
- Persistence mechanisms
- Artifacts left by malware

7. Detections & Report

- Create YARA/Sigma rules
- ATT&CK mapping for technique classification
- Document remediation steps
- Provide containment guidance

Core Competencies (Part I)

1. Behavioral Analysis

- **Lab hygiene**: VM isolation, snapshots for safe rollback
- **Tools**: ProcMon, Sysinternals suite
- **Sysmon telemetry**: detailed system monitoring

2. Static Triage

- **File format analysis**: PE (Windows), ELF (Linux), Mach-O (macOS)
- **Imports/strings**: identify functions and embedded text
- **Entropy & packers**: use DIE (Detect It Easy) to identify packed files
- **Capability hints**: use `capa` tool
- **YARA triage**: quick signature-based identification

Lecture 6 – Reverse Engineering in Politics

3. Code Reverse Engineering

- **Assembly languages**: x86/x64 and .NET IL (Intermediate Language)
- **Disassemblers**: IDA Pro, Ghidra
- **Debuggers**: x64dbg, WinDbg for live stepping through code
- **Analysis**: reasoning about Control Flow Graph (CFG) and data flow

4. De-obfuscation & Evasion Bypass

- **CFG de-flattening**: restore control flow graph
- **Unpack layers**: remove packing/compression
- **Resolve API hashing**: decode obfuscated API calls
- **Defeat anti-analysis**: bypass anti-debug and anti-VM techniques

---

Core Competencies (Part II)

1. Memory & Config Extraction

- **Tools**: Volatility, Rekall
- **Process dumping**: extract running processes from memory
- **Carving**: extract keys, URLs, and data structures from RAM

2. Document & Script Malware

- **File types**: Office/VBA, PDF/JavaScript, LNK files, PowerShell
- **Macro triage**: analyze macros in documents
- **Safe sandboxing**: isolated execution environment

3. Network/C2 Analysis

- **Beacon timing**: identify periodic communications
- **Protocol/TLS fingerprinting**: identify communication patterns
- **URI patterns**: analyze URL structures
- **PCAP-to-IOC pipeline**: extract network indicators from packet captures

4. Reporting & Detections

- **Clear write-ups**: for different audiences (technical/non-technical)
- **Engineerable rules**: practical, implementable detection rules

Goldsmiths
UNIVERSITY OF LONDON

- **ATT&CK mapping**: categorize using MITRE ATT&CK framework

---

Essential Toolchain

Reverse Engineering & Analysis

- **IDA Pro**: industry-standard disassembler
- **Ghidra**: NSA's free disassembler
- **x64dbg/WinDbg**: debuggers for stepping through code
- **dnSpy/dnlib**: .NET decompilers

System Monitoring

- **Sysinternals Suite**: ProcMon, Autoruns, Process Explorer
- **Sysmon**: detailed system event logging

Network Analysis

- **Wireshark**: packet capture and analysis

Memory Analysis

- **Volatility**: memory forensics framework

Detection & Identification

- **capa**: capability detection tool
- **YARA**: pattern matching for malware

Scripting

- **Python**: automation and custom tools

How to Break Into Malware Analysis

1. Build a Lab Environment

- Set up isolated VMs (no bridged network by default)
- Keep meticulous notes
- Use snapshots for safe rollback

2. Learn Foundations

- **OS internals**: understand operating system architecture
- **Assembly/IL**: x86/x64 assembly and Intermediate Language
- **PE/.NET metadata**: understand file formats
- **Windows API**: know common API functions
- **Networking**: TCP/IP, HTTP, DNS fundamentals

3. Practice Regularly

Weekly goals:

- Unpack one sample
- Bypass one anti-debug technique
- Extract one config
- Write one YARA rule

4. Formal Training

GREM (GIAC Reverse Engineering Malware) - SANS FOR610

- Debugging techniques
- Unpacking methods
- Memory analysis
- Network analysis
- Professional reporting

Critical Skills for Success

1. **Technical depth**: assembly, file formats, OS internals
2. **Patience**: malware analysis is time-intensive
3. **Methodical approach**: systematic workflow from triage to report
4. **Tool proficiency**: master the essential toolchain
5. **Documentation**: clear notes and reports for different audiences
6. **Continuous learning**: malware evolves constantly

Lecture 6 – Reverse Engineering in Politics

ATT&CK Framework Integration

- Map observed behaviors to MITRE ATT&CK techniques
- Provides standardized language for threat intelligence
- Helps organizations understand attack patterns
- Enables better defensive strategies

Key Takeaways

1. **Malware analysis is systematic** - follow the workflow: static → dynamic → RE → IOCs
2. **Multiple analysis layers** - combine static, dynamic, and code analysis
3. **Practical outputs matter** - IOCs, YARA rules, and remediation guidance
4. **Tools are essential** - master the core toolchain
5. **Lab safety** - always work in isolated environments
6. **Continuous practice** - regular hands-on work is crucial
7. **Professional certification** - GREM (SANS FOR610) is the industry standard
8. **Document everything** - good notes and reports are critical deliverables

Goldsmiths
UNIVERSITY OF LONDON