Bryan Madewell
IS488 under Professor Karl
Research Week 1 Beginning on 9/13/2024

1. How Autoruns Works (Windows)
Autoruns is a utility that provides a comprehensive view of all auto-starting locations, including registry keys, startup folders, and other mechanisms where programs or services can configure themselves to run automatically on system boot. Understanding autoruns is very important for diagnosing startup issues or detecting malware. Many normal software programs run automatically on system boot, but the issue arises when malware is run upon system boot. This is why I will choose to use Autoruns in my project.

2. Services and Programs (Windows)
A service in Windows is a background process designed to perform functions without user intervention, usually running even before a user logs in. They usually start automatically during system boot. A program, on the other hand, is a user-initiated process with a visible interface. So, a service is generally ran without user interaction, and a program is generally started by a user.

3. Types of Executable Programs
Windows supports various executable formats, the most common being EXE (Executable files), BAT (Batch files), CMD (Command scripts), and PS1 (PowerShell scripts). Each serves different purposes, from traditional programs to automation and administrative tasks.

4. Dynamic Link Library (DLL)
A DLL is a collection of reusable code and data that multiple programs can access. It allows for code modularity, and its dynamic nature means that it is loaded into memory only when needed by an executable program.

5. How Programs Call DLLs
Programs use system calls like `LoadLibrary` or `GetProcAddress` to load DLLs dynamically. This allows applications to share code and use resources efficiently. Relationships between EXEs and DLLs involve defined dependencies where an executable is compiled with references to specific DLL functions.

6. Executable Files, PowerShell, Batch Files Relationship
Executable files (.exe) are direct programs that interact with the OS kernel and while PowerShell and batch files (.ps1, .bat) are scripting languages that rely on interpreters (PowerShell or cmd.exe) to execute commands. Batch files are older, and PowerShell offers more advanced functionality.

7. Verified Programs
A verified program refers to a program signed with a trusted code-signing certificate. This verification ensures that the program hasn't been tampered with and originates from a trusted developer. Verifying involves obtaining a certificate from a Certificate Authority (CA). In order to get a certificate, it must go through an application and approval process.

8. Over-Permissive Services and Programs
Over-permissive services and programs have more privileges than necessary, violating the principle of least privilege. This creates vulnerabilities in systems, as services can be exploited to perform actions in which they are not intended to.

9. LUA Buglight (Concept of Least Privilege)
LUA Buglight is a tool that identifies problems in applications running with standard user rights. It helps developers modify software to conform to the principle of least privilege, ensuring that apps function without administrative rights.

10. Windows Sysinternals (Mark Russinovich)
**There are many, so the first few are the most relevant to my current project, and the PS suite may be useful for powershell scripts I may create. Many Sysinternals seem to be customization options, such as AutoLogin upon startup, blue screens for testing purposes, among other things that people may use to test their system.**
**- Autoruns: Lists all programs configured to run during system boot.**
**- Process Explorer: Monitors real-time processes and their resource consumption.**
- PsExec: Executes processes remotely.
**- TcpView: Views active network connections.**
**- ProcMon: Monitors and logs system activity, including registry, file system, and processes.**
- AccessChk: Verifies permissions on files, registry keys, and services.
**- Sigcheck: Verifies the digital signatures of files.**
- VMMap: Visualizes virtual memory usage by processes.PsExec: Executes processes on remote systems.
**PS Suite:**
- PsFile: Shows files opened remotely via network shares.
- PsGetSid: Displays the Security Identifier (SID) of a computer or user.
- PsInfo: Provides basic system information such as uptime, memory, and installed hotfixes.
- PsKill: Kills processes by name or process ID on a local or remote system.
- PsList: Lists information about running processes, such as CPU usage and memory statistics.
- PsLoggedOn: Shows the users currently logged on to a system.
- PsLogList: Dumps event log records to the command line.

- PsPasswd: Changes the password for a local or remote system.
- PsPing: Measures network latency and bandwidth.
- PsService: Views and controls services on a local or remote system.
- PsShutdown: Shuts down or restarts a computer remotely.
**- PsSuspend: Suspends processes on the system, pausing the processes. Maybe use to suspend things (maybe not).**
- PsTools: A suite of command-line utilities for managing local or remote systems.

11. Windows Home vs. Windows Server
Windows Home is designed for personal use, limiting background services and features related to enterprise management. Windows Server, on the other hand, runs multiple services such as DNS, and Active Directory, focusing resource management and user permissions, allowing programs and services to work freely.

12. Active Directory
In an Active Directory environment, some services can be treated as objects. These objects include properties such as permissions, making them manageable like users or computers within Active Directory. Applications like group policies can control how services behave across the network. Active Directory helps manage things like users, groups, and network resources for Windows.

**Project pitch:**
Maybe scratch autoruns, focus more on processes running.

I will create a software that focuses on enhancing Windows security by analyzing over-permissive services and applying the principle of least privilege. I'll be using tools such as **Autoruns** to ensure that there are no malicious softwares running upon startup. **Process Monitor (ProcMon)** will be used in order to monitor running programs, see what they are trying to interact with, and help identify security concerns. **LUA Buglight** will be used to ensure that every application only has the permissions necessary to serve its purpose, and nothing more than needed. **AccessChk** can be used to ensure that only trusted users or services have the necessary permissions to modify autostart programs, reducing the likelihood of unauthorized alterations. **TcpView** can be used to view active network connections, as some malware may attempt to reach out to unknown IP addresses (IT340 lab). The project will delve into how services and autostart programs can be monitored and restricted to reduce security risks. The analysis will highlight potential vulnerabilities that arise from unnecessary privileges and suggest optimizations to increase the safety of the system.

Incorporating Active Directory, the project will demonstrate how services and applications are managed within an enterprise environment, showcasing the control and restriction mechanisms that can be enforced. By automating this security audit process using PowerShell or custom scripts, the project aims to develop a tool that can properly identify over-permissive services and verifies program certificates, improving system security at all levels.

---

---

RESEARCH FOR THIS WEEK

- Look up processes in attack mitre network
- Look how processes interact with other things (maybe DLL, look at normal things and malicious things and see a difference) looking for a very specific answer (low level, it does this and reaches out…. etc)
    - When shared libraries are invoked, how is it working (literally how is it working, runs .exe and communicates, uses system32, etc.)
- Research when malicious processes are doing bad things, what are they doing (see what bad actors like to do, and set up some rules to stop that)
- Ideas to leverage event log
    - Define rights previously listed (in original research list)
- Tokens and rights stuff
- Focus on the ProcMon thing, and the Galactic Use Case thing from book

NOTES FROM 511.4 ENDPOINT SECURITY ARCHITECTURE BOOK
511.5 AUTOMATION AND CONTINUOUS SECURITY MONITORING
SANS book, GIAC certifications
**CERTIFICATIONS TO GET:**
SEC+
AZURE
SOMETHING CLOUD

- Privileged account monitoring (privileged accounts will always be targeted, but they are necessary)
- Understand and monitor how to limit high-level accounts like admins
- NTFS permissions (page 120) focus on both key data itself, where it is stored, and also focus on areas that are often targeted (two approaches, maybe one leads to another)

- User rights and privileges (rights are just to logon, privileges are anything other than that)
    - They are generally controlled by a group policy
- Controlling user rights can reduce impact of any compromise
    - allow/deny any of the following: local login, remote desktop login, access via the network, logging in as a service
- Controlling privileges can lead to a greater increase in security
- Debugging programs, impersonating as a client, creating tokens, loading drivers, taking ownership, and restoring files/directories (@JasonFossen)
- 13 Sensitive privileges that result in EventID 4672 (have to do with the allow/deny services, they are the same things except the actual privileges)
- Less privilege is more security
- Use ProcMon to reduce privileges is very good
    - Many softwares attempt to take the path of least resistance (poorly coded apps simply take advantage of admin privileges to be easy)
    - App vendors are often lazy, so they don't want to have to write every single permission/process that has to run, if admin privileges are available, it makes everything easier
    - It also results in efficiently produced applications
- Windows (low sodium) password hashes
- Different types of logons
- Access tokens, token impersonations
- Windows Hello (not really relevant to this project)
- Number Six vs Galactica (simulated attack)
    - Attempting to compromise a computer on the galactica windows active directory domain and become the domain admin
    - Attacking system is BASESTAR
    - Target is VALKYRIE, windows 10 system using microsoft defender antivirus
    - The goal of the attack is PEGASUS, the primary domain controller for the GALACTICA domain
    - Attacker view includes tools such as hydra, wimiexec.py, and metasploit
    - Victim view will be shown using windows event logs
        - Password Sprays are used (a few of them)
        - Metasploit attack
        - Metasploit PsExec antivirus logs
        - wmiexec.py (stealthier psexec) evaded defender antivirus, some commands/scripts/etc. are detected by antivirus and EDR (endpoint detection) and some are stealthier (see how the stealthier ones avoid being detected)
        - The stealthy attack was not detected in the antivirus logs

- You can discover privileges with commands like “whoami -all” so see if this requires something like admin privileges
- Plan.exe metasploit
- Make sure things like meterpreter sessions are killed (attempt to access system was blocked, but session was still running, so the bad actor was still able to get into the system and disable the antivirus)
    - Be aware that stopping the attempt to attack is not the only thing, there may be lingering sessions or loose ends still around
- Obviously, RDP is something that is a huge vulnerability
- Registry Keys, malware tends to maintain a C2 connection, and maintain persistence after reboot
- PowerShell stuff