Sterilized Persistence Vectors (SPVs): Defense Through Deception on Windows Systems

Nicholas Phillips
Towson University
Nphill5@students.towson.edu

Dr. Aisha Ali Gombe
Louisiana State University
Aaligombe@lsu.edu

Cyber 2022

Author

Nicholas Phillips

Vulnerability Researcher and Malware Analyst for Parsons
PhD Candidate - Towson University

Nicholas Phillips is a Vulnerability and Malware Researcher for Parsons Corporation. Multi-time published researcher into better means of defense against compromise. Former US Navy, he has been on the front lines of computer science for over 15 years.

Active with multiple organizations, including UPE, IEEE, and AFECA.

He is currently in the final phases of his Ph.D. at Towson University, specializing in Malware.

01 Outline

Outline

Outline (Continued)

Evaluation (Defense Against Malware)

Evaluation (Reversion Testing)

Evaluation (System Performance)

Evaluation (White Listing)

Conclusion

Questions

02

Problem Statement

Malware: From Beginning to Current

“It is time to wake up and smell the Mutating Hash! Signature Based Malware Detection is Dead.”

—James Scott
Senior Fellow, Institute for Critical Infrastructure Technology

Malware Constant Evolution

- Malware authors evolve their practices
- 64 confirmed Zero Days Utilized in 2021

Security One Step Behind

- Innovations such as Secure Boot have been found vulnerable
- Malware authors have been even able to forge security certificates

Focus Generally on Exterior Not Interior

- Major security tools attempt to identify infections prior to infection
- Very little is researched into areas where malware sets its hooks

One Grouping Remains Constant

- Malware Persistence Mechanisms
- Key Entries May Change, But Regions are the same

Malware Utilized For Defensive Strength

- Malware Constantly Uses Security Features to Strengthen Their Attacks
- Why Not Utilize Their Strength to Add Defensive Power?

Introduce Sterilized Persistence Vectors

- SPVs are formed in Executable matching with Benign Rootkit
- Defense By Deception can prevent malware infections via appearing currently infected or revert persistence changes launched by malware

“People’s computers are not getting more secure. They’re getting more infected with viruses. They’re getting more under the control of malware.”

—Avi Rubin

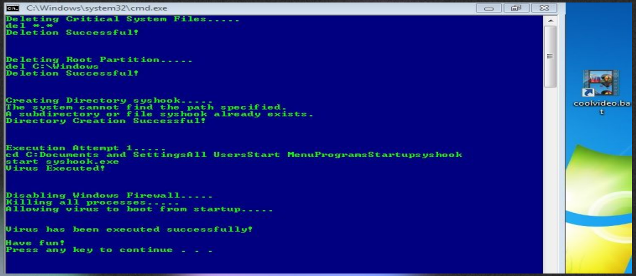
03

Longitudinal Study

Understanding the current threat of malware

Malware Longitudinal Study

- Malware samples were pulled from multiple malware analysis sites and analyzed per persistence, complexity, and security.
- Malware continues to evolve but even across generations their persistence remains fairly consistent.
- Blocking persistence limited the amount of system resources needed for defense



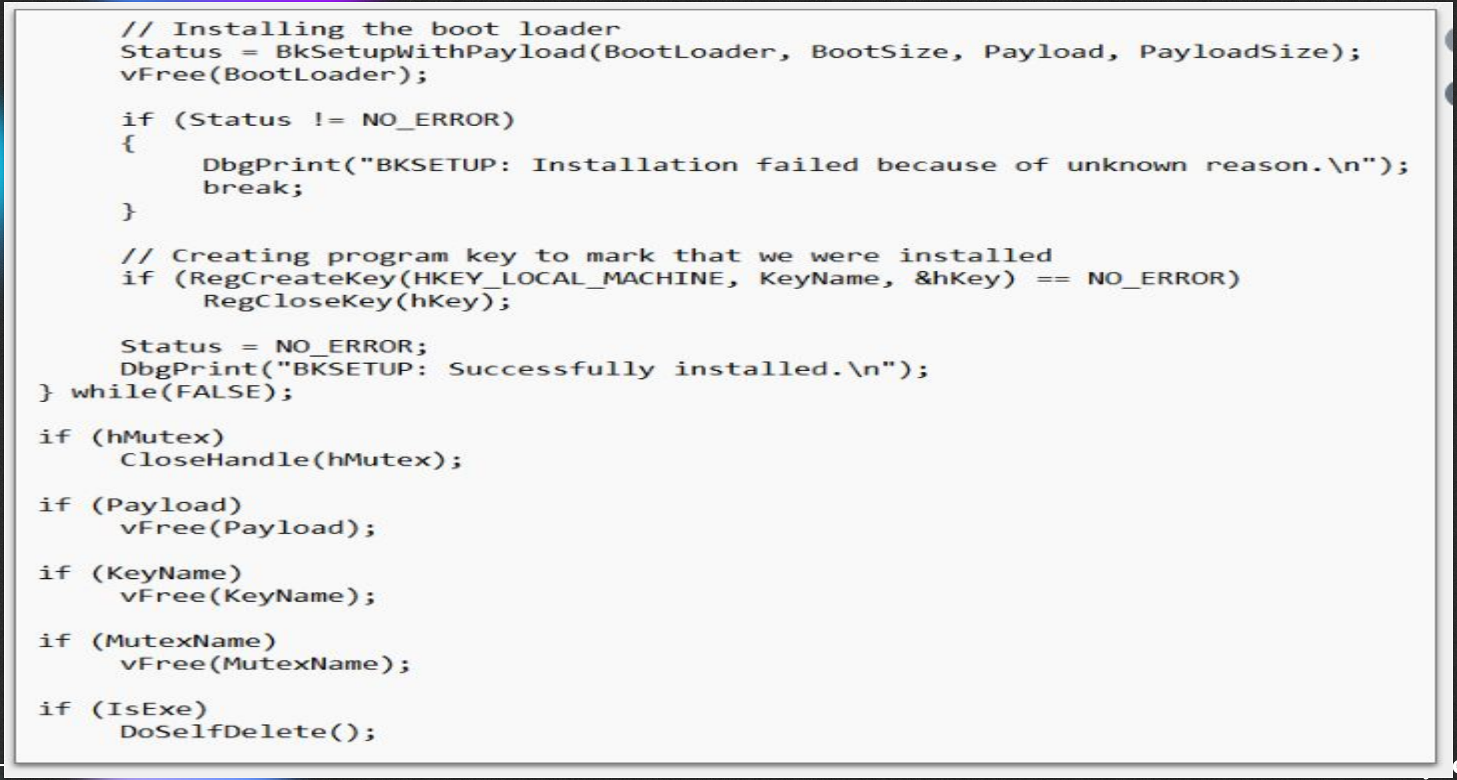
```
Deleting Critical System Files.....
del *.*
Deletion Successful!


Deleting Root Partition.....
del C:\Windows
Deletion Successful!


Creating Directory syshook.....
The system cannot find the path specified.
A subdirectory or file syshook already exists.
Directory Creation Successful!


Execution Attempt 1.....
cd C:Documents and SettingsAll UsersStart MenuProgramsStartupsyshook
start syshook.exe
Virus Executed!


Disabling Windows Firewall.....
Killing all processes.....
Allowing virus to boot from startup.....

Virus has been executed successfully!

Have fun!
Press any key to continue . . .
```

04

Sterilized Persistence Vectors

Defense Through Deception

Extracted PV



```
        // Installing the boot loader
        Status = BkSetupWithPayload(BootLoader, BootSize, Payload, PayloadSize);
        vFree(BootLoader);

        if (Status != NO_ERROR)
        {
            DbgPrint("BKSETUP: Installation failed because of unknown reason.\n");
            break;
        }

        // Creating program key to mark that we were installed
        if (RegCreateKey(HKEY_LOCAL_MACHINE, KeyName, &hKey) == NO_ERROR)
            RegCloseKey(hKey);

        Status = NO_ERROR;
        DbgPrint("BKSETUP: Successfully installed.\n");
} while(FALSE);

if (hMutex)
        CloseHandle(hMutex);

if (Payload)
        vFree(Payload);

if (KeyName)
        vFree(KeyName);

if (MutexName)
        vFree(MutexName);

if (IsExe)
        DoSelfDelete();
```

SPVs

- Extracted from multiple different malware samples
- Independent code generated for over 800 persistence vectors
- Matching PVs across the samples were paired down to matching of the base, with unique keys placed as variables
- White listing generated from same process being conducted on legitimate programs

05

SPVExec Deployment

Building and Deploying “Benign” Malware

SPVExec

- Form a benign rootkit of the SPV and implements persistence elements called SPVExec.

- Additional persistence scanning mechanisms, were added to the code to overwrite non-whitelisted persistence modifications.

- Malware functionality deployed a FAT32 file system within the bootstrap code section.
  - Used for SPV library, whitelisting, and the SPV Defense base code.

- The data remained encrypted, utilizing a 256-bit key to protect against registering on scans.

- SPVExec implemented as single Windows executable program loaded alongside the essential boot files at system startup.

06

Evaluation (Persistence)

Only Strong if it Remains, Like Malware

Defensive Persistence

Defensive measures must maintain its own persistence

Evaluation

- SPVExec code executed on system, and a power cycle was performed
- Memory dump collected off of system
- Memory image was processed by Volatility Memory Framework with the following plugins: psxview, malfind, ldrmodules, apihooks, dlldump, procdump, and threads
- Information showed the existence of SPV defensive processes running on system

07

Evaluation (Defense Against Malware)

Does it Protect Against the Threat?

Defense Against Malware Persistence

Defensive measures must stop malware from gaining persistence on a system

Evaluation

- For this experiment, SPVExec run in two unique phases.
  1. Determined if malware identified SPVs as similar malware.
  2. If legitimate programs, such as Antivirus, saw the SPVs as a benign.

- 1000 attempted infections of showed no signs of infection on SPV defended system.

- 15 AVs saw the SPVs as benign code

08

Evaluation (System Performance)

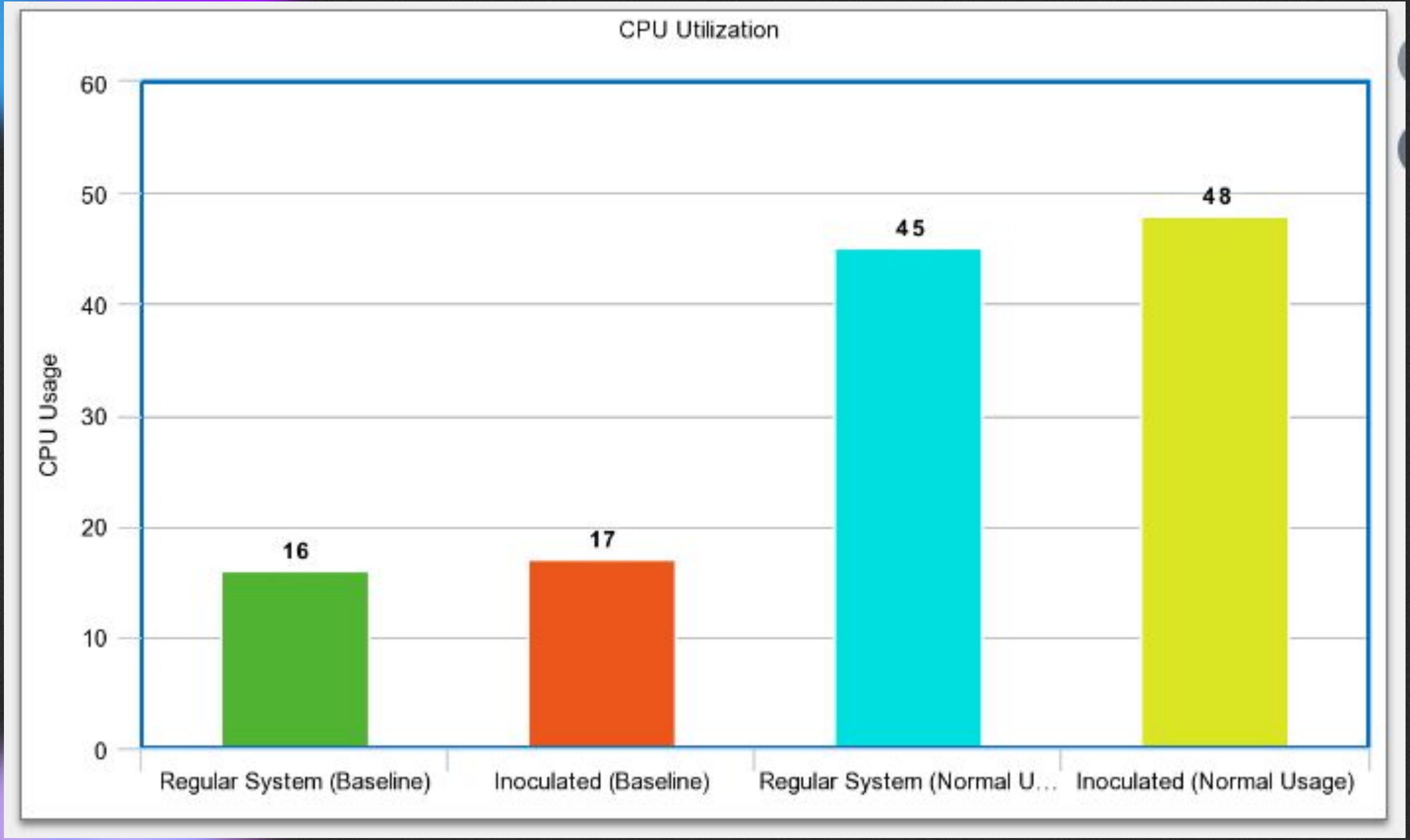
Is There Significant Impact on the System?

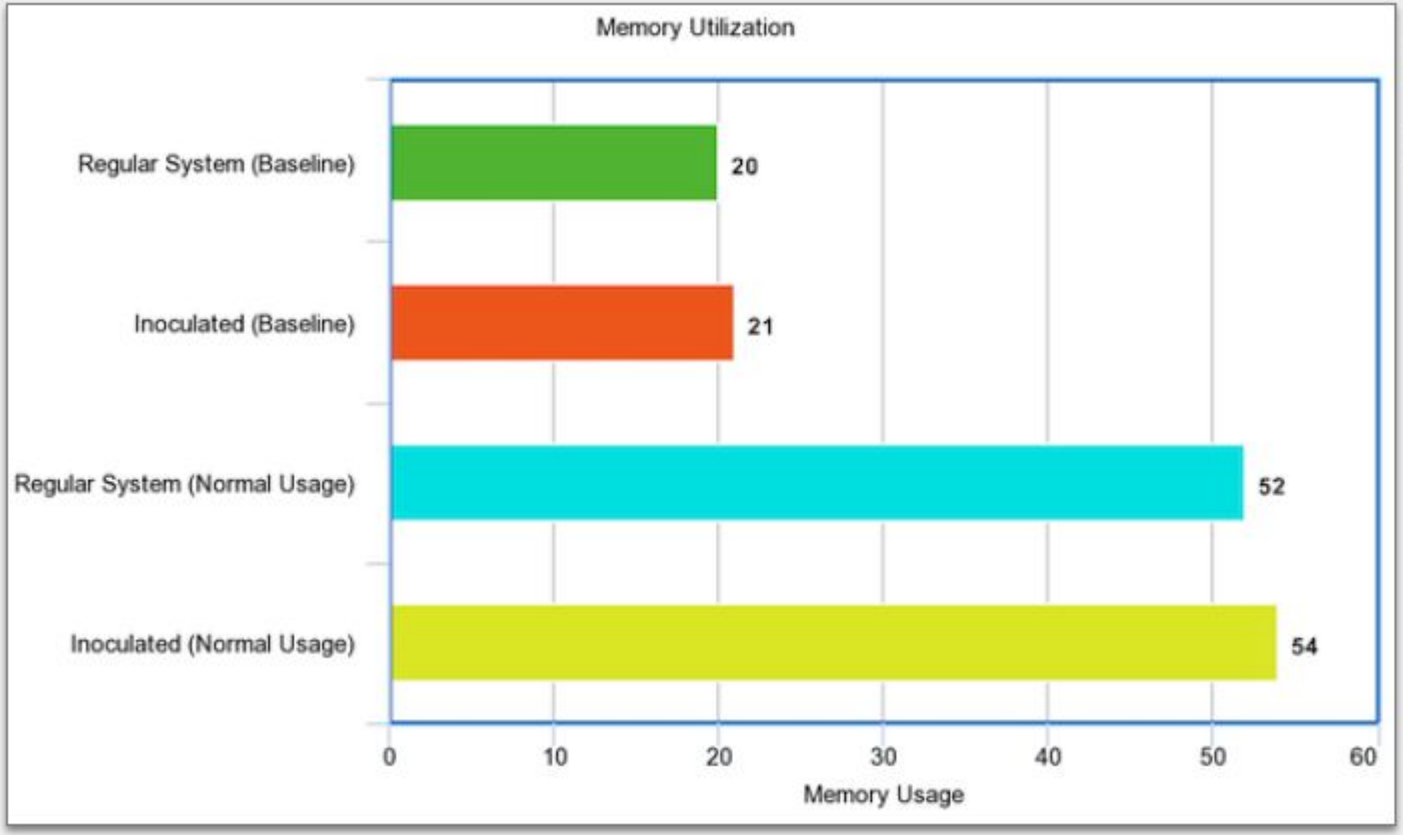
System Performance Analysis

Defensive measures must be limited in the impact to the system.

Evaluation

- Evaluated effectiveness of our approach particularly the impact on memory and CPU utilization.
- Utilization was recorded in two separate instances to obtain a baseline for the pre- and postdeployment system.
- Applications opened: two Microsoft Word documents, instance of Google Chrome, and one instance of the Windows file structure.

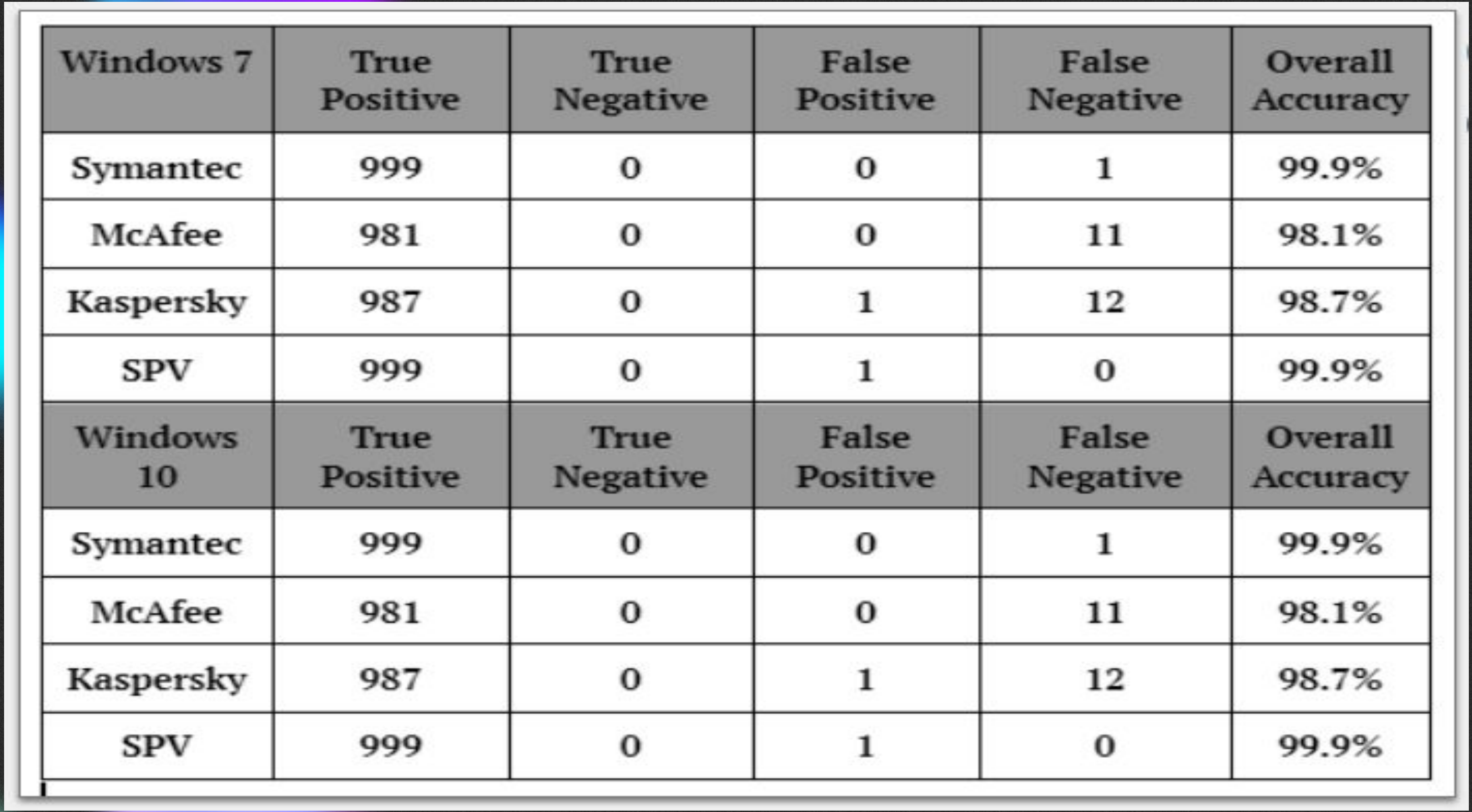

CPU Utilization
CPU Usage
60
50
40
30
20
10
0
16
17
45
48
Regular System (Baseline)
Inoculated (Baseline)
Regular System (Normal U...
Inoculated (Normal Usage)


Memory Utilization
Regular System (Baseline)
20
Inoculated (Baseline)
21
Regular System (Normal Usage)
52
Inoculated (Normal Usage)
54
0
10
20
30
40
50
60
Memory Usage

09

Evaluation (Regular Testing)

Do SPVs Defend Similar to Antivirus?

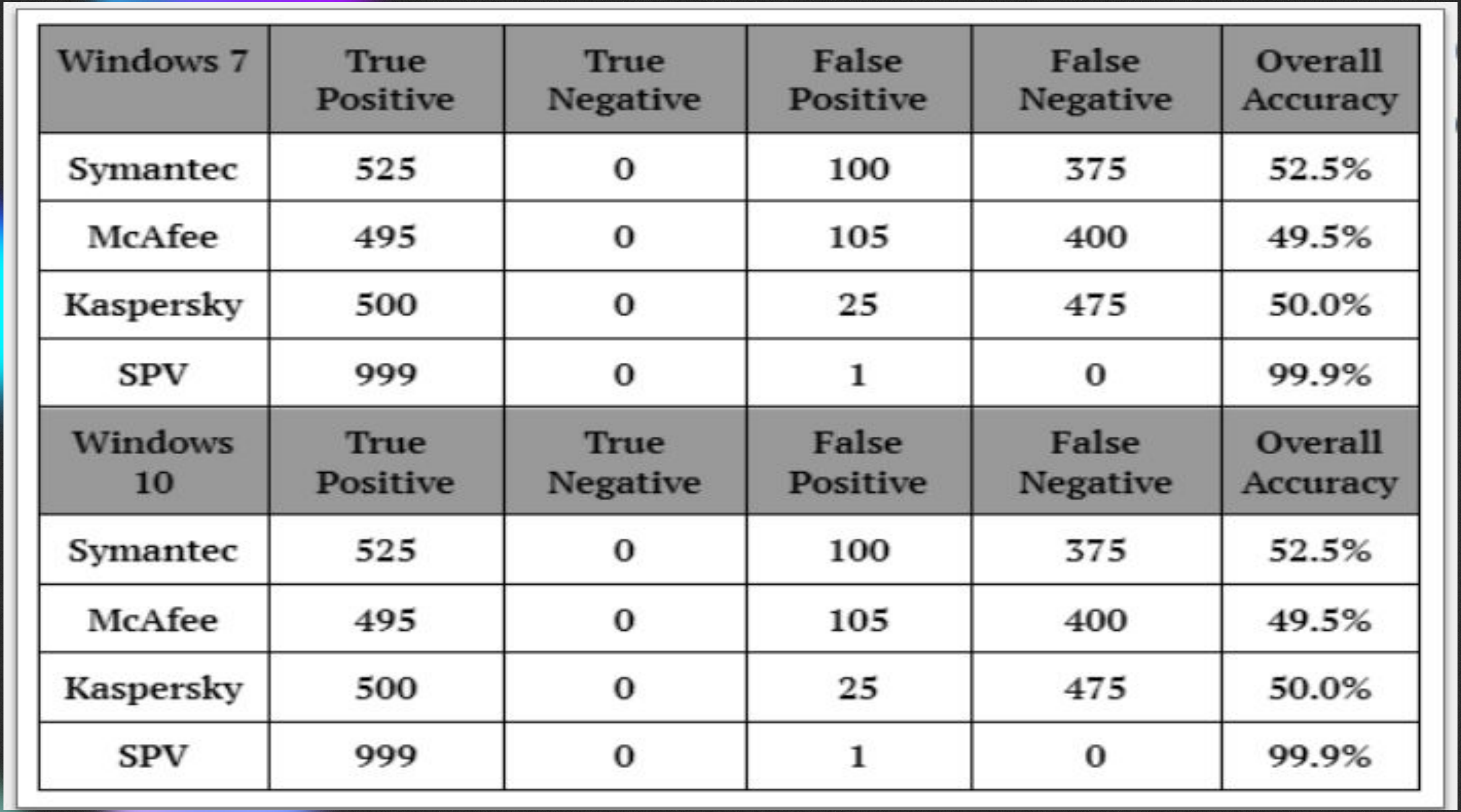


Windows 7	True Positive	True Negative	False Positive	False Negative	Overall Accuracy
Symantec	999	0	0	1	99.9%
McAfee	981	0	0	11	98.1%
Kaspersky	987	0	1	12	98.7%
SPV	999	0	1	0	99.9%
Windows 10	**True Positive**	**True Negative**	**False Positive**	**False Negative**	**Overall Accuracy**
Symantec	999	0	0	1	99.9%
McAfee	981	0	0	11	98.1%
Kaspersky	987	0	1	12	98.7%
SPV	999	0	1	0	99.9%

10

Evaluation (Reversion Testing)

Does It Defend Against Unknown Threats?



Windows 7	True Positive	True Negative	False Positive	False Negative	Overall Accuracy
Symantec	525	0	100	375	52.5%
McAfee	495	0	105	400	49.5%
Kaspersky	500	0	25	475	50.0%
SPV	999	0	1	0	99.9%
Windows 10	**True Positive**	**True Negative**	**False Positive**	**False Negative**	**Overall Accuracy**
Symantec	525	0	100	375	52.5%
McAfee	495	0	105	400	49.5%
Kaspersky	500	0	25	475	50.0%
SPV	999	0	1	0	99.9%

11

Evaluation (White Listing)

Can the Good Still Gain Persistence?

Deployment of Legitimate Persistence

Defensive measures are not useful if they cannot allow legitimate programs to gain persistence

Evaluation

- PyCharm, Visual Studio, BitRise, Atom, BlueFish, CodePen, Crimson Editor, Eclipse, Komodo Edit, and NetBeans were installed on defended system.
- Each instance was able to obtain persistence and able to operate normally

12

Conclusion

What Are The Final Results?

SPVs: Defense By Deception

Proved to be a strong defensive measure against various malware attacks on Windows Systems

- While not infallible, these persistence scanning elements could be added as an additional layer or decoy in a fully deployed defense-in-depth methodology.
- Found an exponential trend for complexity and stealthiness, with samples only becoming more adapted to overcome the security tools in place to protect systems

Required Modifications For Change To Linux/Unix

With the commonality of the persistence vectors across malware, there is potential for this to be converted into a security tool.

Scanning capability found in more modern samples that will not complete infection if other infection is present.

13

Questions

?

Thank you for your time

Do you have any questions?

nphill5@students.towson.edu