APPLICATION WHITELISTING: APPROACHES AND CHALLENGES

Himanshu Pareek, Sandeep Romana and P R L Eswari

Centre for Development of Advanced Computing, Hyderabad, India
{himanshup, sandeepr, prleswari}@cdac.in

ABSTRACT

Malware is a continuously evolving problem for enterprise networks and home computers. Even security aware users using updated security solutions fall into trap of zero day attacks. Moreover, blacklisting based solutions suffer from problems of false positives and false negatives. From here, idea of Application whitelisting was coined among security vendors and various solutions were evolved with same underlying technology idea. This paper provides the details about design and implementation approaches and discusses challenges while developing an effective whitelisting solution.

KEYWORDS

Malware, Application Whitelisting, Enterprise Network

1. INTRODUCTION

With increasing number of new malware [1] it is becoming difficult for traditional antiviruses following blacklisting principle to detect them. To address this problem new approaches (among which application whitelisting is one) are being developed. Application Whitelisting [2] is a methodology where user chooses a set of applications to run from rather than blacklisting applications as done by traditional antivirus software. The user might be a single PC user and chooses the application set for one computer or it might be an administrator who chooses a large set to run on the network. Whenever any application starts on a given computer, solution checks whether this file is present in whitelist or not. If not then running a process using this file is denied. Figure 1 compares three basic approaches for malware prevention.

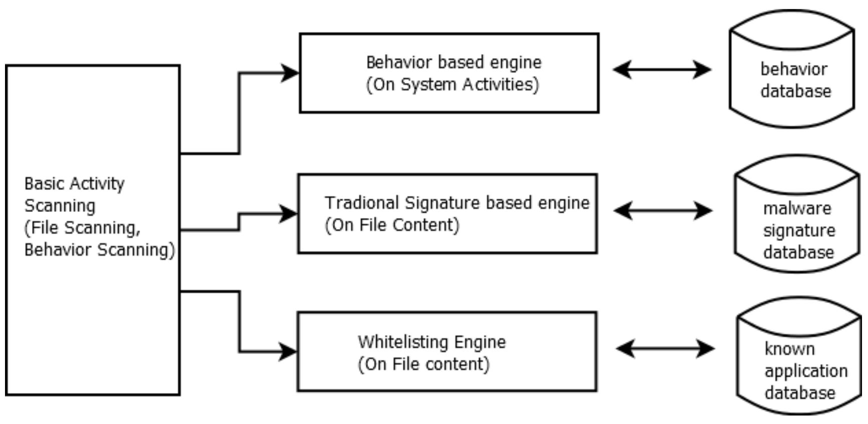

Basic Activity Scanning (File Scanning, Behavior Scanning)
Behavior based engine (On System Activities)
behavior database
Tradional Signature based engine (On File Content)
malware signature database
Whitelisting Engine (On File content)
known application database

Figure 1: Primary approaches for malware prevention

Application whitelisting solutions use an underlying kernel module to scan the basic activity in system. Scan engine compares application details with the database. General architecture for centralized application whitelisting solution is presented in section 3. Approaches for implementing a whitelisting solution is discussed in section 4. Usage of various parameters to uniquely identify and whitelist an application is discussed in section 5. Challenges associated with the effective implementation of application whitelisting solution are discussed in section 6.

2. RELATED WORK

Application Whitelisting looks promising at the first sight but researchers have argued its feasibility and practicality. Kurt Wismer [3] mentioned that determining whether a program is good or bad is itself an intractable problem. Dr. Vesselin Bontchev [4] discussed the problems of global and local whitelists. Jim Beechey [5] describes the advantages of whitelisting and attacks possible on such kind of systems and concludes that though application whitelisting is not a panacea but still required to counter the evolving malware attacks. In this paper, we bring out two different approaches of application whitelisting enforcement and lists out issues with application whitelisting which should be taken care while deploying.

3. GENERAL ARCHITECTURE FOR A CENTRALIZED APPLICATION WHITELISTING SOLUTION

Centralized Application Whitelisting Solution generally uses client-server architecture as given in figure 2. Application Whitelisting Server helps to create and maintain database of application whitelisting policies for various client machines of the network. Based on the policies, whitelist and notifications would be sent to each of the network client machines. Enforcement agent at client machine would allow execution of only those applications which are in the whitelist and blocks all other applications. Client agent would update the log details to the server at regular intervals of time for further analysis.

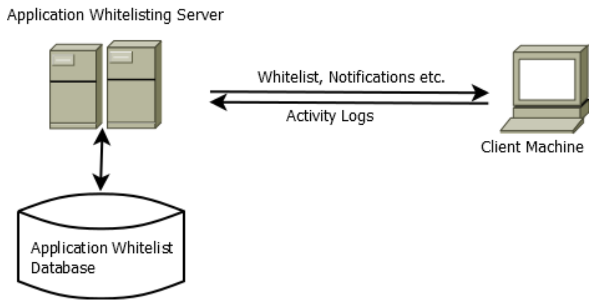

Application Whitelisting Server
Whitelist, Notifications etc.
Activity Logs
Client Machine
Application Whitelist Database

Figure 2: General Architecture for Centralized Application Whitelisting

4. TWO APPROACHES FOR WHITELISTING

In order to enforce the application whitelist at client machine, there are two general approaches as described below.

1. Analyse the executable file at runtime during process creation

In this approach, client enforcement agent will have an intercepting module which looks for the creation of new processes on the system. In case of any new process creation event, intercepting module communicates the details to another module which scans and checks whether this executable file exists in the application whitelist of that client machine. In case if it is not a whitelisted application, this module indicates to the user that this application is not allowed on that machine. Figure 3 gives pictorial view of this approach. There are various methods for implementing the intercepting module of client enforcement agent using process creation event. List of methods used for detecting the process creation event on Windows platform are:

1. User mode hooking of CreateProcess using DLL injection (or CreateProcessAsUser and CreateProcessWithLogon can also be hooked). For such injections detour library can also be used [6].
2. User mode hooking of NtCreateSection using DLL injection.
3. Kernel mode hooking of NtCreateSection [7].
4. Using PsSetCreateProcessNotifyRoutineEx (Windows Vista onwards) [8].
5. Utilizing software restriction policies (provided with Microsoft windows).

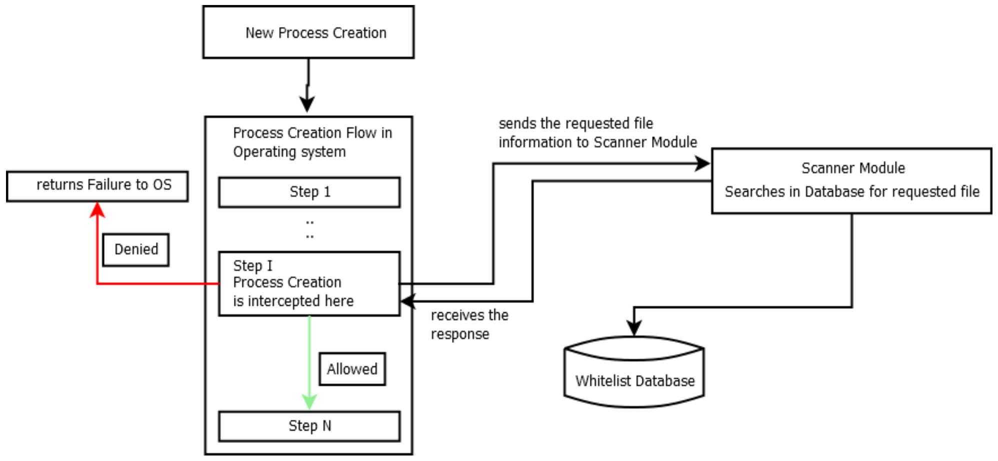

New Process Creation
Process Creation Flow in Operating system
Step 1
Step I
Process Creation is intercepted here
Step N
Allowed
Denied
returns Failure to OS
sends the requested file information to Scanner Module
receives the response
Scanner Module
Searches in Database for requested file
Whitelist Database

Figure 3: Application Whitelisting Enforcement during Process Creation

2. Scan every file of file system and verify the details of executable file against application whitelist of the client machine

In this approach, client enforcement agent first scans executable files and verifies them against the application whitelist of that machine. In case if the file is not found in the application whitelist of that machine, then it can be either deleted or quarantined.

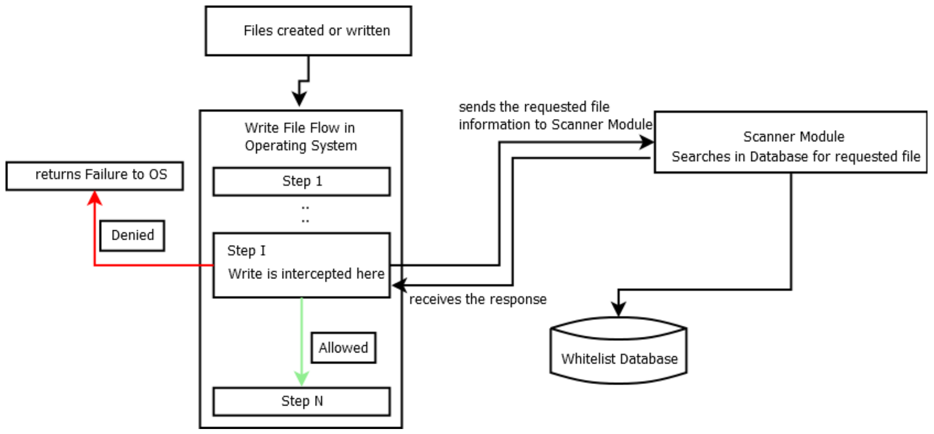

Files created or written
Write File Flow in Operating System
Step 1
Step I
Write is intercepted here
Step N
Allowed
Denied
returns Failure to OS
sends the requested file information to Scanner Module
receives the response
Scanner Module
Searches in Database for requested file
Whitelist Database

Figure 4: Application Whitelisting Enforcement during File Write

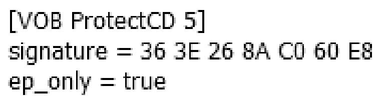
However quarantine is a better option as the user gets the files back for any future requirements. In this approach, intercepting module should monitor and verify the details of any new executable file entering into the machine. This approach is pictorially shown in figure 4. This module cannot rely on file extensions, but should verify based on file format.

5. PARAMETERS FOR IDENTIFYING AN APPLICATION

In order to identify whether the application is a whitelisted application or not, parameters of the executable file need to be verified with the details provided in application whitelist database. An application can be identified easily with executable name and path but these details may vary on different client machines and also applications can be replaced. For example winword.exe if executed can be allowed or denied based on the executable name and path as provided in application whitelist database. But hacker can simply put some malicious executable with the same name and path in the file system. Since the name and path are matching malicious executable will be whitelisted and allowed for execution. These parameters combined with Hash and Publisher name would be a better approach for verifying the details of executable files. Therefore four fields would be maintained for each entry in the application whitelist database.

$$Wn = \{File\ name, Path, SHA512, Publisher\ Name\}$$

Application signature is another parameter which can be used in application whitelisting. This approach is similar to the method of detecting malware using their signatures [9]. For example consider a signature taken from userdb.txt [10].



```
[VOB ProtectCD 5]
signature = 36 3E 26 8A C0 60 E8
ep_only = true
```

Figure 5: Signature which can identify a PE packed with VOB Protect

The signature shown in figure 5 identifies any executable which is packed using VOB ProtectCD 5. The signature shows seven bytes. If an executable file also contains same sequence of seven bytes, scanning module will report success. Moreover this approach also states that the search must start at entry point of the executable. Sometimes these signatures can contain wild card characters also to skip a single or multiple bytes.

6. CHALLENGES

Though application whitelisting solutions are able to provide security from zero-day attacks, there are various challenges in order to effectively implement these solutions.

1. **Building Trusted Application Database**
   This is the main challenge while offering an application whitelisting solution. There are two options to deal this challenge. First option is give the responsibility of building trusted application database to administrator. He will be responsible for identifying, verifying and whitelisting the applications. The other option is to depend on verified database of trusted applications maintained by third party and administrator only has to choose and approve them.

2. **Update and Patch Handling**
   Integration of Application Whitelisting with patch management is another important challenge which the solutions should address. When an application is kept in the whitelist, its identification parameters (either hash or signature) are stored. When the update or patch is installed, application identification parameters will change, resulting into whitelisting application being denied. This problem must be addressed by integrating with a patch management solution and monitoring such update activities.

3. **Publisher Certificate Forgery**
   With the current technology one can verify and also check the validity of digital signature of an application but the challenge in application whitelisting solution is verification of stolen certificates. There are many cases of malware using stolen certificates [11]. This type of malware will clear the verification process, if security policy allows software from a particular application vendor.

4. **Performance Degradation with DLL Whitelisting**
   When application whitelisting is extended to include DLL (Dynamic Link Libraries) whitelisting, there is considerable effect on performance when loading an application. Though DLL whitelisting provides the added advantage of better security but considering the performance degradation problem it's a tough decision for application whitelisting vendors to consider including DLL whitelisting in their solution.

5. **Virtualization**
   Virtualization aides in administration, deployment and development tasks but also exposes security threats. These include security threats to hypervisor, virtual OS and data loss. Moreover if virtualization is not configured correctly in the network, users may bypass the application control and device control. If any virtualization software is white listed then administrator must make sure that application whitelisting client software is installed in the virtual operating systems. Otherwise this will enable the user to run the unauthorized applications. Moreover, if users can simply install their own new virtual machines then it will bypass application whitelisting. This also enables the users to run the unauthorized applications and devices.

7. CONCLUSION

Application whitelisting is evolving as a promising methodology to address the zero-day threat. This methodology allows only whitelisted applications for execution, whereas all other applications are blocked. This paper lists the design and implementation approaches for application whitelisting solutions. Also discusses various challenges, in order to effectively implement application whitelisting solutions.

REFERENCES

[1] Malware Statistics. AV-Test. http://www.av-test.org/en/statistics/malware/
[2] Dave Shackleford, "Application Whitelisting: Enhancing Host Security", http://www.sans.org/reading_room/analysts_program/McAfee_09_App_Whitelisting.pdf, October 2009
[3] Kurt Wismer. The rise of whitelisting. Available from http://anti-virus-rants.blogspot.com/2006/03/rise-of-whitelisting.html
[4] Dr Vesselin Bontchev. The dark side of whitelisting. Available from http://www.virusbtn.com/virusbulletin/archive/2007/08/vb200708-whitelisting#citation.5

[5] Jim Beechey. Application Whitelisting: Panacea or Propaganda. http://www.sans.org/reading_room/whitepapers/application/application-whitelisting-panacea-propaganda_33599

[6] Alex Abramov, "API Hooking with MS Detours", http://www.codeproject.com/Articles/30140/API-Hooking-with-MS-Detours, Oct 2008

[7] Anton Bassov, "Hooking the native API and controlling process creation on a system-wide basis", http://www.codeproject.com/Articles/11985/Hooking-the-native-API-and-controlling-process-cre, October 2005

[8] Microsoft Dev Center - Hardware, "PsSetCreateProcessNotifyRoutineEx routine", http://msdn.microsoft.com/en-us/library/windows/hardware/ff559953(v=vs.85).aspx, January 2012

[9] Nwokedi Idike and Aditya P. Mathur, "A Survey of Malware Detection Techniques", Technical Report, Purdue University, 2007

[10] BoB / Team PEiD, "Signature Database", http://code.google.com/p/reverse-engineering-scripts/downloads/detail?name=UserDB.TXT, February 2011

[11] Jarno Niemelä, "It's signed, therefore it's Clean, right?" CARO 2010

Authors

Himanshu Pareek has around six years of experience in developing and design of security solutions related to small sized networks. He has research papers published on topics like malware detection based on behaviour and application modelling.

Sandeep Romana has nearly 5 years of experience in the field of research and developing of security software for desktop's and small networks. His areas of research include behavioral malware detection and application whitelisting

Mrs P.R.L. Eswari is currently working in e-Security team at Centre for Development of Advanced Computing (C-DAC) Hyderabad. Her areas of interest include End Point Security, Network Security, Operating Systems, Linux, Data Structures, Algorithms and Design.