International Journal of Enterprise Computing and Business Systems

ISSN (Online) : 2230-8849

http://www.ijecbs.com

Vol. 2 Issue 1 January 2012

WINDOWS-BASED APPLICATION AWARE NETWORK INTERCEPTOR

Ms. Shalvi Dave[1], Mr. Jimit Mahadevia[2], Prof. Bhushan Trivedi[3]

[1] Asst.Prof., MCA Department, IITE, Ahmedabad, INDIA

[2] Chief Architect, Elitecore Technologies Ltd., Ahmedabad, INDIA

[3] Director, MCA Department, GLSICT, Ahmedabad, INDIA

Abstract

To protect important network resources from threats or malicious access, one has to implement security at both network level and host level. Any IDS or IDPS achieves this by normally using a set of pre-defined rule-sets. However, existing evasion techniques have become more intelligent by invading the network or a host by effectively pre-judging the rules defined and used by IDS/IDPS. In addition, the false positives rate is usually high for such systems, which also hampers overall network performance. The paper aims to describe how a novice approach helps identifying applications as an attacker or a victim by intercepting system calls on Windows. Our system introduces three main modules: Network Interceptor, Event Collector and Administrative Server. Together, they identify an attack with help of NIDS along with the name and version of the attacker and victim applications that we can protect in future. This paper first describes how Network Interceptor intercepts system calls and classifies an application as an attacker and a victim. An event log is also maintained for such attacker applications within the network. The event log contains information such as Source and destination IP, application name and version, timestamp etc. It is also used to generate administrative reports. The administrator to take corrective measures for such applications uses these reports.

Keywords: Event Collector, Layered Service Provider, Network Interceptor, Winsock

I. Introduction

The Network Based IPS (NIPS) is also denoted as in-line proactive protection. It is also proactive because it intercepts all network traffic and inspects for suspicious behavior and code. We require intercepting live data because we need to send it to IDS engine to get it verified. This interception has to be real time along with dropping or blocking functionality.

We classify two types of applications running in a network: Client application and Server Application. Whenever a client application in the network requests for any service outside the network, it may become vulnerable to attacks from servers running outside the network. In addition, when any service provided by a Server application within the network is requested by an outside application, it may also launch an attack on server application. Apart from this, any vulnerable or infected application, client or server can possibly make attacks, to an application outside the network.

We have designed and implemented a Windows-based Network Interceptor (NI) using Layered Service Provider of Winsock 2 in C language, which intercepts live data at run-time when any application launches an attack or when any application is being victimized. The Network Interceptor intercepts system call functions to retrieve actual network data along with application information. It captures socket function calls such as read, write, sendto, recvfrom etc. NI sends application information to SURICATA engine for classification of an attack, and if attack is found, it logs the attack-related data. It then generates event log including application related information using Logging Agent [6]. Network Interceptor along-with IDS logging agent sends this event log to Event Collector. The Event Collector stores this event log in a database. This event log helps to generate administrative reports. These reports provide various attack related statistics. In future, the Network Interceptor can query the administrator for new applications or existing connections. The administrator, in turn, will match the application information with the quarantine database and send the decision back to Network Interceptor. Using this decision, it would prevent attacks. Thus, Network Interceptor, as a subsystem, can be used as a part of IDPS.

II. Related Work

IDPS solutions are widely used nowadays for protecting both hosts and network infrastructure from attacks of all kinds including DOS attacks, malware, worms, Trojans, and application specific threats [1]. An IDPS detects a potential security breach, drops the malicious traffic, logs the information and sends alarm to the administrator. In this section, we describe two systems: SNORT and Bluebox HIDS.

A. Snort

Snort, open-source IDS, uses a set of pre-defined signatures [2]. Snort sends attack information on syslog, which contains only network and attack related information. It does not get stored into database for future analysis. In addition, Snort also does not have prevention capabilities on Windows platform, which is very much required to achieve security effectively.

B. Bluebox, policy driven HIDS

Attack log analyzer of bluebox HIDS correlates network logs and system logs [3]. There are two major problems with this approach. First, correlation analysis of large data is quite impossible. In addition, as events are generated from different subsystems, there are chances that correlation analysis might not be precise. Second, Policy defined in bluebox enforces rules controlling application access to system resources like socket [3]. In addition, pre-defined policies can define overall system's behavior but cannot change it during run-time. It has to be automatic and run-time to achieve better security.

Need for a Windows-based Network Interceptor:

In our solution, we have tried to provide solutions to the above-mentioned problems. In our approach, we store events into the database so it can be used for future analysis. We also store application information like application name and version. It can be used in future to quarantine application or to apply any hot fixes if available. We also achieve precision because we include run-time event information.

In addition, Network Interceptor hooks system functions and works on real data so as soon as it detects an attack. Hence, in future, it can drop that data to achieve real time prevention. Therefore, an IPS or IDPS solution can also use Network Interceptor.

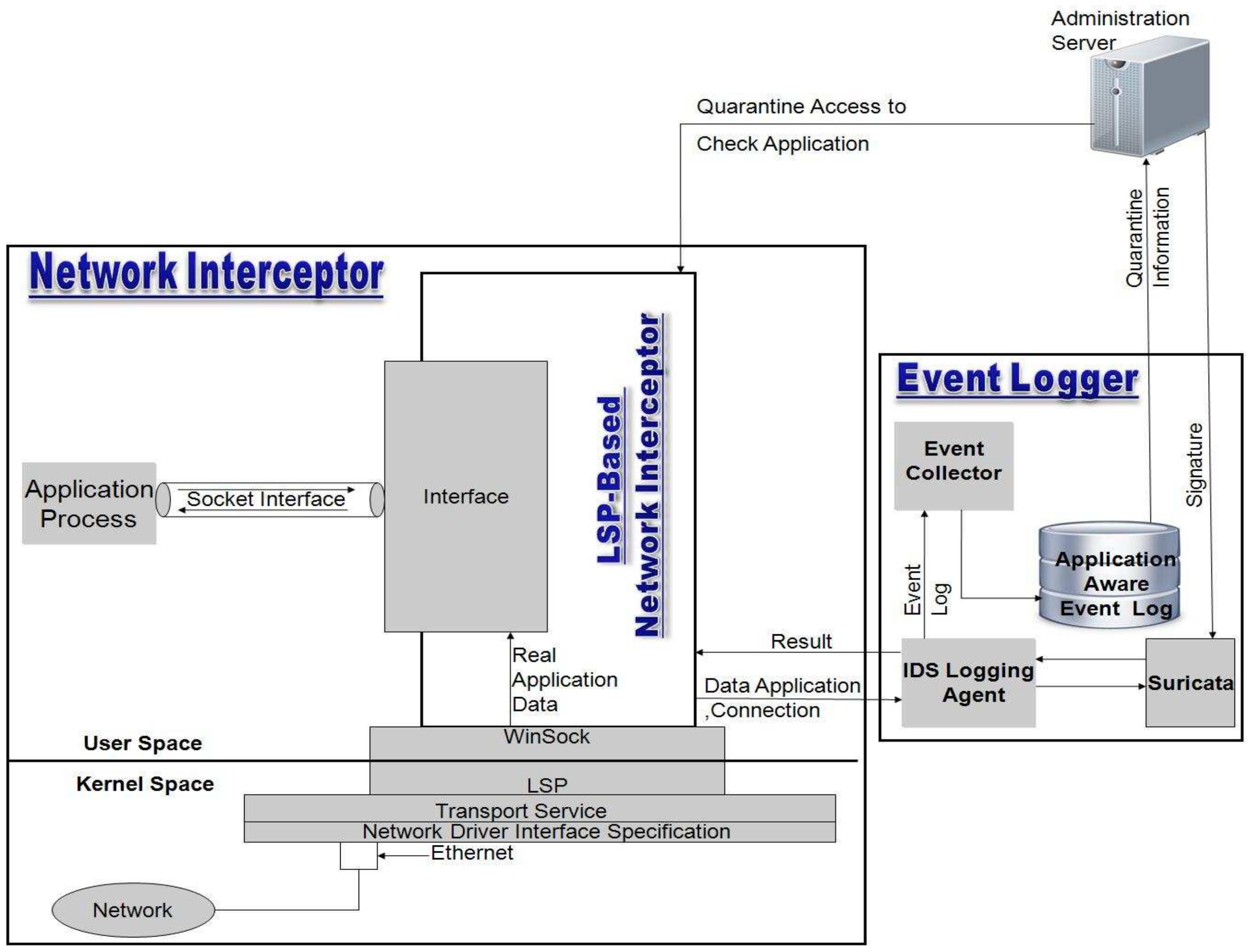


FIG. 1 System Architecture

Figure 1 describes how Network Interceptor is implemented on Windows platform. We have used Postgre SQL database to store event logs. We have divided the entire system into two main functioning areas: Network Interceptor and Event Logger. The Event Logger is already detailed in our previous

International Journal of Enterprise Computing and Business Systems

ISSN (Online) : 2230-8849

http://www.ijecbs.com

Vol. 2 Issue 1 January 2012

paper [6]. In this paper, we describe working of system, followed by how Network Interceptor hooks the system call and access the real data. How Network Interceptor with the help of Logging Agent and Suricata together identify attack classified as: Inside client attacking outside server, inside client being victimized by outside server, inside server being victimized by outside client and inside server attacking outside client has been already covered in our previous paper [6]. We also describe how Network Interceptor along-with Logging agent and Suricata detects attacks real time and generates event-log with help of Event Collector. In the end, the results of all four class studies are shown.

III. Working

Our system consists of three modules: Network Interceptor, Event Collector and Administration Server. Logging agent and network interceptor are installed on each host on the network. Event Collector and Administration server are centralized. Logging agent and network interceptor communicates with each other via TCP. The above diagram shows the actual working of our Windows-based Network Interceptor.

To provide network services or to access network resources, applications uses native socket interface provided by Windows. Network interceptor has been instantiated by Winsock for every socket. To get access of this network data, Network Interceptor hooks this socket interface. As it hooks directly in socket, it also gets the socket information (source IP, source Port, destination IP, Destination port, protocol) and details of application (application name, version etc.) which has opened that socket.

When any application sends or receives the network data via socket, Network Interceptor's hooking function is called. Details of hooking mechanism have been described later in this paper under Network Interceptor section. In this blocking function, NI gets the control on data and holds processing of the same. Network Interceptor, then with the help of Logging agent and Suricata, inspects the network data to detect possible attack on or by this application. In case of attack, Suricata provides name of the attack, severity of the attack and type of the attack, which we have briefed later in this paper under

Suricata. We have described details of event and quarantine log in our previous paper [6]. Network Interceptor can use this information in future to prevent attacks or drop existing connection.

The following section describes how Network Interceptor identifies attacker and victim applications run- time. It retrieves real data by intercepting function calls, sends this information to logging agent. Logging agent uses Suricata for attack identification and classification and sends result back to Network Interceptor. The following class studies describe how Network Interceptor classifies attacks:

A. Classification Of Attacks

This section describes how Network Interceptor intercepts application aware information real time. Then role of each component is explained with context to the entire system architecture.

For a proof of concept, we have implemented Apache web server and IE application on Windows platform. We have taken four windows hosts in our lab setup. Two hosts are configured as Internet Hosts and other two are configured as Inside hosts. One inside host is running with IIS web server. One Internet host is running with Apache web server. Another Internet host is deployed with IE to make request to internal web server. For storing application aware event log, we are using Postgre SQL database. We have classified signatures in two different kinds of rule-sets using flow direction already specified into signature.

Attack On Client

When any client accesses any service from server, there are high chances that an attack can be launched on client by the program running on remote server. This attack is client specific. It is due to any known vulnerability of specific client version. The client –side attack can be inbound as well as outbound depending on the direction of connection.

A. Outbound Attack (Inside Client being victimized): One of the desktop machines tries to access site hosted on the server on Internet using Internet Explorer. If this site or server is being compromised then it can launch an attack on the internet explorer application. One such type of attack is “Heap Buffer Overflow”. If the version running of Internet Explorer is vulnerable to this attack then it is being victimized by such an attack. Network Interceptor gets the name and version of application along-with criticality of attack. In future, it can use this for prevention. The following diagram shows the actual working of the same:

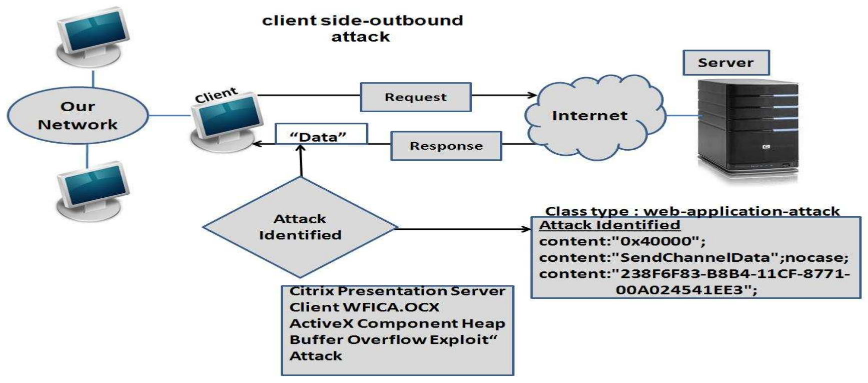

client side-outbound attack
Server
Client
Request
Our Network
Internet
"Data"
Response
Attack Identified
Class type : web-application-attack
Attack Identified
content:"0x40000";
content:"SendChannelData";nocase;
content:"238F6F83-B8B4-11CF-8771-00A024541EE3";
Citrix Presentation Server Client WFICA.OCX ActiveX Component Heap Buffer Overflow Exploit" Attack

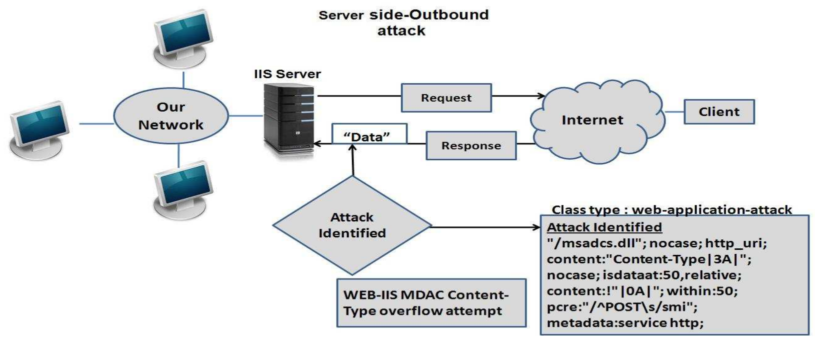
FIG.2. Client-Side Outbound Attack

B. Inbound Attack (Inside server attacking on outside client): In this scenario, IIS server is running in our network. Some client from the internet tries to access the site using Internet Explorer. If our server is infected or compromised, then it can launch an attack on the remote client’s IE. Our Network Interceptor intercepts and detects this attack and sends the log to central event collector via logging agent.

Attack On Server

When anyone requests any service from the server, he can also land an attack along with the request. If running application server has any known vulnerability, services can suffer due to attack. Again, depending on direction of connection, attack can be inbound as well as outbound.

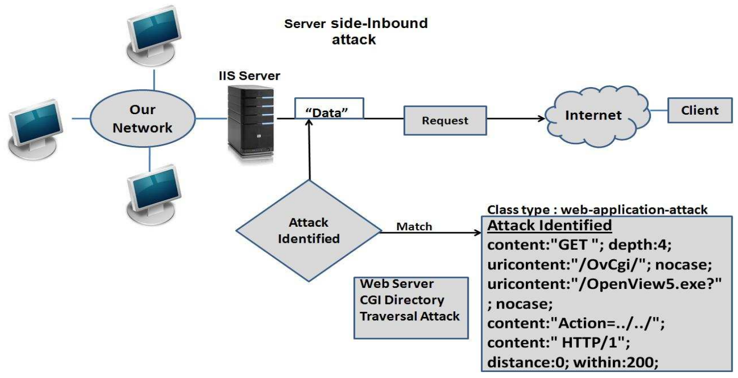
A. Outbound Attack: A server is hosted on the internet. One of the desktop machines in our network tries to launch an attack using an IE application. This application launches “Remote Code Execution Attempt” attack on to remote server. Since it is an attack, which is generated from within the network, information is also stored into quarantine database.


Server side-Outbound attack
IIS Server
Our Network
Request
Internet
Client
“Data”
Response
Attack Identified
WEB-IIS MDAC Content-Type overflow attempt
Class type : web-application-attack
Attack Identified

```
"/msadcs.dll"; nocase; http_uri;
content:"Content-Type|3A|";
nocase; isdataat:50,relative;
content:!"|0A|"; within:50;
pcre:"/^POST\s/smi";
metadata:service http;
```

FIG.3. Server-Side Outbound Attack

B. Inbound Attack: In this case, IIS server is running in our network. Client attempts directory traversal on our server. The following figure shows how an inbound attack occurs in the network:


Server side-Inbound attack
IIS Server
Our Network
“Data”
Request
Internet
Client
Attack Identified
Match
Web Server CGI Directory Traversal Attack
Class type : web-application-attack
Attack Identified

```
content:"GET "; depth:4;
uricontent:"/OvCgi/"; nocase;
uricontent:"/OpenView5.exe?"
; nocase;
content:"Action=../../";
content:" HTTP/1";
distance:0; within:200;
```

FIG.4. Server-Side Inbound Attack

International Journal of Enterprise Computing and Business Systems

ISSN (Online) : 2230-8849

http://www.ijecbs.com

Vol. 2 Issue 1 January 2012

Functionality of Network Interceptor

As described in previous section, Network Interceptor with the help of logging agent and Suricata inspects traffic to check whether exploit has been found. If found then agent sends an event to central event collector. If the attack is classified as in-bound attack, Network Interceptor sends an alarm to the administrator. If the attack is classified as out-bound, application information is stored in the quarantine database.

Network Interceptor can be used by any IDS/IPS for detection and prevention. The reason is it intercepts live application data at run-time. Using quarantine information stored in the database, it can prevent new attacks or drop existing connections. Also, using quarantine information, administrator can apply security hot-fixes if available. Thus Network Interceptor can be used by both IDS as well as IPS.

As mentioned before we are using Suricata in our solution. Following is a brief description of how Suricata is used by our system.

IV. Suricata

Our Network Interceptor inspects real data with the help of IDS logging agent and Suricata. Suricata uses pre-defined signature rule-set to identify an attack. Network Interceptor sends data along with application and connection information to Suricata via logging agent. Therefore, Suricata does not need to track the connection for TCP Reassembly. After vulnerability scanning, Suricata sends result back to logging agent. Result includes information about attack if detected along with its severity.

Suricata uses standard rule-set available from emerging threats. IDS signature rule-set has to be updated from time to time to detect new threats. The logging agent contacts the administration server to check availability of new signature in the signature database. If new signature is found, logging agent also updates Suricata with new signature.

As already mentioned, we have designed and implemented three modules for our solution. The first one is Network Interceptor that is described below:

International Journal of Enterprise Computing and Business Systems

ISSN (Online) : 2230-8849

http://www.ijecbs.com

Vol. 2 Issue 1 January 2012

V. Network Interceptor

Intrusion prevention systems are considered extensions of intrusion detection systems because they both monitor network for malicious activity. The main differences are, unlike intrusion detection systems, intrusion prevention systems are placed in-line and are able to actively prevent/block intrusions that are detected [4]. To achieve this one need to sit inline in to the connection flow and monitor them for malicious activity. As soon as one found it should also drop that connection and log the event. Our subsystem Network Interceptor can be used for prevention of new attacks or dropping existing connections. Network Interceptor sits inline of the application socket data, it can be used to prevent application from different attacks by dropping malicious connections. We have described technical approach to achieve the same in following paragraph.

Network Interceptor can use two different techniques on Windows: One is using NDIS Framework, which works at Kernel Level and intercepts raw network data. Another approach is using WINSOCK Hooking, which works at intermediate level and intercepts socket calls and data. We have chosen WINSOCK hooking as it also provides application information along with socket call interception.

There are different methods available to implement WINSOCK Hooking like replacing WINSOCK related DLLs, DLL Hijacking and writing LSP Driver. Out of all these approaches, Microsoft has recommended writing LSP driver. All other approaches may lead to compatibility issue with different version of Winsock DLLs. For this reason, we have chosen to implement Transport Layered Protocol using WINSOCK 2 SPI Framework.

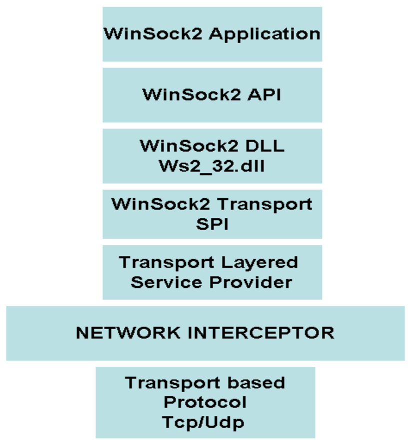


FIG. 5. Layering Using Network Interceptor

Network Interceptor is being used to extend an existing transport service provider by implementing a protocol stack as a services, which supply functions that set up connections, transfer data, exercise flow control, error control, and so on. Network Interceptor is implemented as a standard Windows DLL. To use a LSP SPI Architecture of Windows we need to register our DLL with SPI Framework's API called WSCInstallProvider [1]. Once we register this DLL, all transport SPI functions implemented by Network Interceptor are made accessible to ws2_32.dll via the LSP's dispatch table using a callback function mechanism. Therefore, Network Interceptor can intercept Winsock 2 functions before they are processed by ws2_32.dll and it can override the functions. We are interested in functions, which are being used by application to send or receive the data. Such functions are normally send, recv, sendto, recvfrom, etc. which help in intercepting the data.

When any application wants to use socket in windows it must initialize WINSOCK library. As soon as an application initiates the connection for the first time, our Network Interceptor DLL also gets initialized along with it. During the time of initialization, we supply pointers of our callback functions, which get registered into dispatch table. Therefore, when application calls any socket functions like send, recv, create, etc., it first calls our function. Our function gets the access of socket handle and transmitted data. It sends this data to SURICATA for inspection and waits for its result. The result determines future decisions of Network Interceptor. Upon receipt of the result, it would either ALLOW or DENY this connection and achieve prevention capabilities. Thus, Network Interceptor is tested as a sub-system to achieve and implement run-time prevention. The idea behind testing NI for both intrusion detection and prevention is to develop an application aware IDPS.

VI. Event Collector

Using the concept of Producer- Consumer mechanism, we divide the Event Collector into the three sub-systems: Event Receiver, Queue and Database Plug-in.

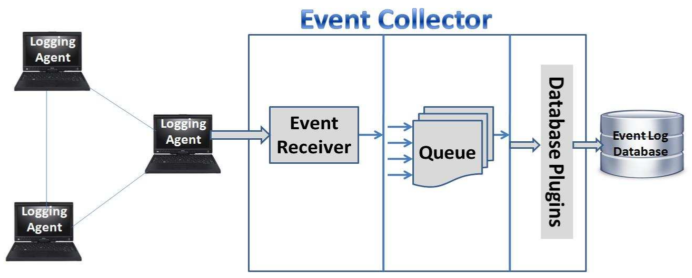
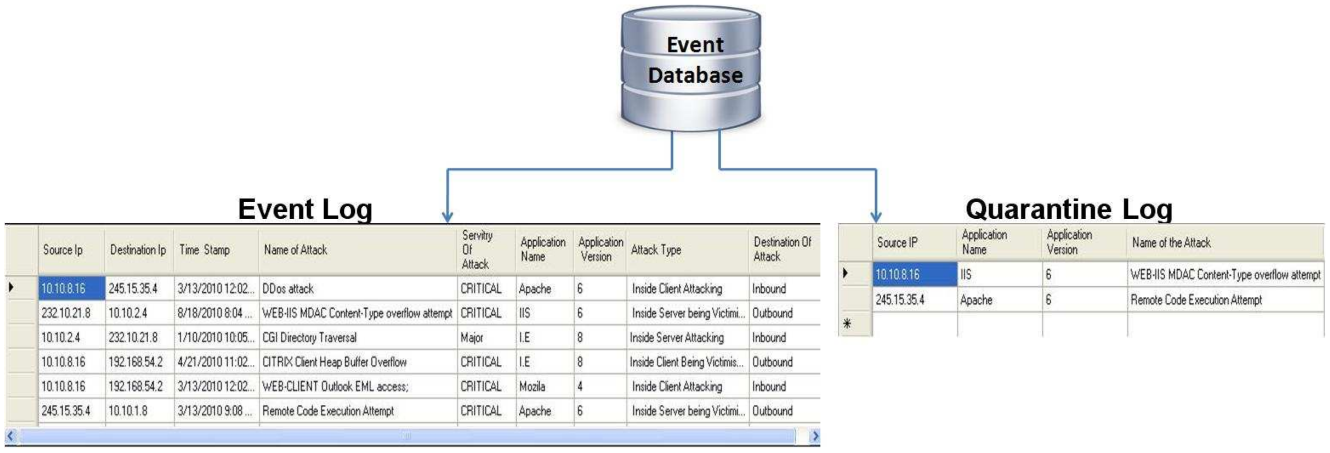


FIG. 6 Event collector

Figure 6 describes the three sub-systems. We have described a detailed explanation for Event Collector in our paper on Event Logger [6]. The logging agent use UDP protocol to send event log to the Event Collector. Event collector receives this log and inserts it in the queue. The database plug-in will fetch this log from the queue and store it in the database.



	Source Ip	Destination Ip	Time Stamp	Name of Attack	Servitry Of Attack	Application Name	Application Version	Attack Type	Destination Of Attack
▸	10.10.8.16	245.15.35.4	3/13/2010 12:02...	DDos attack	CRITICAL	Apache	6	Inside Client Attacking	Inbound
	232.10.21.8	10.10.2.4	8/18/2010 8:04 ...	WEB-IIS MDAC Content-Type overflow attempt	CRITICAL	IIS	6	Inside Server being Victimi...	Outbound
	10.10.2.4	232.10.21.8	1/10/2010 10:05...	CGI Directory Traversal	Major	I.E	8	Inside Server Attacking	Inbound
	10.10.8.16	192.168.54.2	4/21/2010 11:02...	CITRIX Client Heap Buffer Overflow	CRITICAL	I.E	8	Inside Client Being Victimis...	Outbound
	10.10.8.16	192.168.54.2	3/13/2010 12:02...	WEB-CLIENT Outlook EML access;	CRITICAL	Mozila	4	Inside Client Attacking	Inbound
	245.15.35.4	10.10.1.8	3/13/2010 9:08 ...	Remote Code Execution Attempt	CRITICAL	Apache	6	Inside Server being Victimi...	Outbound

	Source IP	Application Name	Application Version	Name of the Attack
▸	10.10.8.16	IIS	6	WEB-IIS MDAC Content-Type overflow attempt
	245.15.35.4	Apache	6	Remote Code Execution Attempt
*				

FIG. 7. Event Log

The application name and version helps the administrator to find out whether any new security hot-fixes are available for the application. If one found, the administrator applies this hot-fix on the application. If there is an out-bound attack, and if Suricata classifies the attack as Critical, then application name and version

is stored in a quarantine database. Using this quarantine information, Network Interceptor can block connections run-time and quarantine applications if their information matches with quarantine database.

VII. Administration Server

Various reports are generated from the event information stored in the database by Event Collector. They are described in our paper on Event Logger [6]. The administrator can use this information to either block or quarantine this application.

When administrator logs into the admin server, he gets alerts if any unacknowledged quarantine application has been found in application quarantine database. Administrator needs to acknowledge this alert, if he chooses to remove this application, from application quarantine database, then next time the application can acquire network resources. Administrator can also choose to apply any new security hot-fixes to application if available. In the later case, as application information has been changed, due to security hot-fix, it is considered as a new application and it can acquire network resources.

Conclusions & Future Extensions

From the above-mentioned technique, we can conclude that Network Interceptor and Event Logger are pre-requisites, which helps to improve security for network resources with application aware information. Administrator can watch the list of quarantined application and take a corrective action if implemented as IDS or IDPS. It can also automate some functions for using the same NI for prevention deploying it with either IPS or IDPS. An administrator or automated solution can
remove the application from the quarantine list or can inspect the host that has landed an attack. This server will also work as a signature distribution server. All IPS engines will connect to this server and download the signature updates.

We can further extend this design to achieve run-time prevention and drop existing connection when Network Interceptor detects an attack. It also helps reduce false positives.

References

[1] Layered Service Provider of Winsock2 available at http://www/microsoft.com/msj/0599/LayeredService.aspx

[2] Snort functioning available at www.snort.org

[3] Suresh Chari and Pau-chen Cheng: ACM Transactions on Information and System Security, Vol. 6, No. 2, May 2003. *BlueBoX: A Policy-Driven, Host-Based Intrusion Detection System*

International Journal of Enterprise Computing and Business Systems

ISSN (Online) : 2230-8849

http://www.ijecbs.com

Vol. 2 Issue 1 January 2012

[4] Intrusion Prevention Systems available at http://en.wikipedia.org/wiki/Intrusion_Prevention_system

[5] Shalvi Dave, Bhushan Trivedi, Dashang Trivedi, International Conference on Computer Modelling & Simulation, *Simulation Of Security Agent Using Anomaly Based Detection and VLAN Steering*

[6] Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, submitted to International Conference on Computer, Communication and Electrical Technology, *Application Aware Event Logger*