Eraser: An Exploit-Specific Monitor to Prevent Malicious Communication Channels

Abhishek Singh

August 26, 2004

Abstract

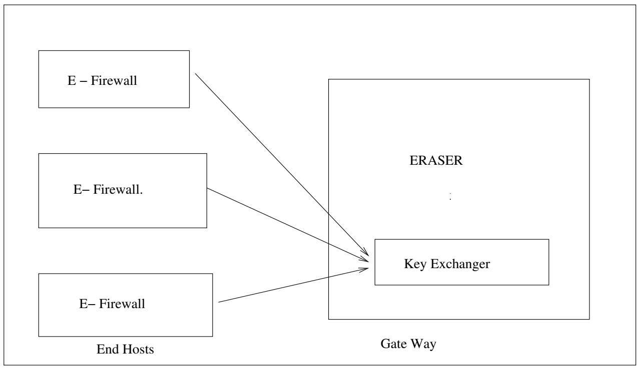
Prevention of malicious communication channels has been an important issue in building secure networked system. Malicious Communication channels can be established by using header fields which follow particular semantics or by using data fields, which do not follow any particular semantics. There have been many research directions to prevent the malicious flow of information in the case of header fields. This paper presents the design of a system which, has been designed exclusively to prevent the malicious covert channels using data fields. The proposed system consists of two parts. The first part is a firewall, which is referred to as E-firewall. The E-firewall runs on the end hosts, while the Eraser runs at the Gateway. Eraser is a rule or policy based system, which checks for malicious content in the payload. Besides storing the trust metrics of the applications,the E-Firewall also stores the dependencies amongst the applications. This storage of dependencies offers incremental advantage over the existing firewall by providing information about the flow of information amongst the applications inside the E-firewall.

1 Introduction

Network security has been one of the most challenging issues that, modern organizations face. Malicious agents like worms can compromise the system and can send the malicious information outside the network. Malicious insiders, another important threat, can steal the sensitive information and send it to the outside network.

A malicious communication channel can be established by using the header fields which follow particular semantics or by the data fields which do not follow particular semantics. The payload of echo_request, echo_reply and the sequence number of unfragmented IP packets are some of the header fields, which have been used for establishing a malicious communication channel. A number of approaches have been explained to prevent the malicious communication channel using header fields. Some of them include, enforcing semantic consistency amongst the unused header fields by using active mapping [11] or by the enforcing the stateless model in the kernel [13] [14]. However, not much research efforts have addressed the malicious communication channel using data fields. The proposed research presents the design of a monitor, which has been designed to prevent the malicious flow of information in the data fields.

The paper presents the design and architecture of Eraser, which has been designed to prevent the malicious flow of information in the data fields. The paper presents the pragmatic challenges of the design and implementation of Eraser. Eraser comprises of two parts: the first part is a firewall (referred to as E-Firewall) that runs on the end



Architecture of the Eraser. E–Firewall runs at the end hosts mahcines and the Eraser runs at the Gateway

Figure 1: Architecture of the System showing Firewall and the Monitoring System

hosts and the second part is called a E-monitor, which runs at the gateway and checks for the malicious information in the payload.

The applications running on the end hosts can be classified into two parts. Trusted applications and untrusted applications. The trusted applications running on the end hosts can further be classified into two parts trusted mutable application and the untrusted immutable applications. Trusted immutable applications are the applications whose integrity has been verified by the vendor and will be streaming trusted data. For example audio server will be streaming songs whose integrity has been verified by the company. Integrity of the song can be stored in the TPM and the trusted audio server before streaming the song can compare the stored hash value of the song with the already stored hash value in the TPM. There exists another class of trusted applications called as mutable trusted applications. Even though the integrity of the application has been verified, they still can contain malicious information. For example MS Word even though the integrity of the word is intact, the doc document can still contain some hidden information. In the proposed system to prevent the malicious flow of information in the data fields, our first contributions is the design and implementation of the E-firewall. A conventional distributed firewall [2] rests on three notions:

1. Policy language specifying the type of connections permitted or prohibited.
2. Number of system management tools such as Microsoft SMS or ASD.
3. IPSEC network-level encryption, mechanism for TCP/IP.

In contrast to all the above, instead of using conventional distributed firewall we propose E-firewall, which uses an identity_stamp for authenticating the packets from trusted immutable applications. The identity_stamp protocol used for authenticating the packets from trusted immutable packets by the E-firewall is light weight as compared to the mechanism used by the distributed firewall.

On the end hosts, applications often access other applications. For example MS word can use excel, excel can access the internet. Our second contribution is the concept of storing the dependencies amongst the applications. Current modern day firewalls only store the hash of the trusted applications. Besides storing the hash the E-firewall also stores the dependencies amongst the applications. These dependencies help to decide the flow of the information amongst the applications.If the integrity of some application is compromised, then from the dependencies it can be inferred what other applications might have received the sensitive information from the compromised application. This in turn helps to monitor other applications for the sensitive information.

Our third contribution is the design and implementation of the monitor Eraser, which checks for the malicious information in the payload. Eraser makes use of trusted computing to achieve the defensive design principles. The policy language used by Eraser is less complicated than Bro [10].

The paper is divided into the following subsections. In the subsection 1.1 we talk about the threat model, subsection 1.2 talks about the potential damage, subsection 1.3 discusses about the related work. Section 2.0 discusses the characterictics of a good design. Section 3.0 discusses the system model and its set of assumptions. Section 4.0 discusses the modules of the system. In section 5.0 we talk about the analysis and the results of the experiments. In section 6.0 we discuss the out come of the experiments and finally in section 7.0 we talk about the research contributions and future work.

1.1 Threat Model

A malicious insider has been the greatest threat to organizations, but the techniques to stop a malicious insider are often time consuming. In the case of the FBI spy Robert Hanssen [4], his disruptive activities were not detected and stopped for over a decade. During this, period he distributed top secret information to the KGB. His success was facilitated by various forms of steganography and other undercover techniques to communicate and send the malicious information.

In the respect of establishing malicious communication channel, Simmon [3] presents the concept of a subliminal channel. Simmon describes a subliminal channel as the one where the hidden data piggybacks on an innocuous-looking legitimate communication channel, whose existence is undetectable. By definition, steganographic carriers are subliminal channel since the communication appears to be innocent, but really has the ulterior information embeded below the threshold of perception.

Active wardens have been an area of postulation since Simmon introduced the Prisoner's Problem in 1983. Simmon presented Alice and Bob as prisoners that collectively wish to plan their escape. However since they are in separate areas of prison, all their communication must pass through the warden Willy. If Willy sees any attempts at secret communication in their messages, he will styme their efforts by not allowing them to communicate in near future. Thus Alice and Bob must use a subliminal channel to communicate their escape plan without altering Willy. Since Willy knows that Alice and Bob may wish to communicate secretly, he must carefully analyze all the correspondence between Alice and Bob, but must do so without perceptively altering their message or incurring a noticably time delay.In this context Simmon defined a subliminal channel as a channel whose very existence is undetectable. The proposed model is designed for an high-security environment where the network is not a free channel but is instead frequently moitored or restricted against the unauthorized usage. For the proposed outbound channel we are assuming that the outbound traffic is not encrypted.

The proposed channel is for a high risk security threat where the computers are monitored for the malicious activities by active and passive wardens. Integrity of the applications are also periodically monitored.In this respect we the integrity of the applications streaming trusted data are periodically monitired or ensured by using access control. This can be achieved by either enforcing the strict access control policies on the applications streaming trusted immutable data or by periodically monitoring the integrity of the applications streaming trusted immutable data.

In such a threat model, an adversary will try to send the malicious information by using various forms of malicious communication channel. He can establish malicious communication channel by either using various forms of steganographic carriers. One of the attempt to establish the malicious communictaion channel can be to send the malicious channel by manipulating some bits in the images or the audio files.

Besides establishing a malicious communication channel there can be other ways to extract the information from a given site such as physically stealing the data from the computers. The proposed system does not address this issue.

1.2 Potential Damage

A malicious communication channel can either be a high bandwidth communication channel or it can be a low bandwidth communication channel. We are inspired from the following incidents to address the issue of high band width communication channels:

• If 500 million packet leave per day, assuming that the malicious insider can control the timing of a packet to get 1 bit of data out per packet, the site could lose over 26 GB annually. If a malicious inside can manipulate 8 bytes in each packet, the site could loose over 4 GB daily. [1]

• In the recent past there was a concern of Al Qaeda using various forms of steganographic techniques to establish the malicious communication channel. [15] [12]

These incidents serve as a motivation for the design and implementation of Eraser which, can checks and remove the malicious information in the data field.

1.3 Related Work

Jonson [8] [7] has done extensive work in identifying signatures for specific steganographic techniques. By closely monitoring the artifacts left from several commercial products, he noticed several distinguished traits for many commercial products. Since all the commercial techniques modify the carriers in some way he was able to document many of these signatures. However, his result was confined to the commercial steganographic packages and malicious insiders would not opt to use such public techniques in their covert transfer. Fridrich et.al [6] discusses a simple technique to detect hidden information in the least significant bits of images by closely observing the number of close colors in the images. While their technique work relatively well for large hidden messages, small-embedded messages produce an error rate up to 40%.

Provos [16] attempted to find images containing steganography on the Internet. He downloaded 3 million JPEG images from Ebay and USEnet, and performed several tests to attempt to determine if they had embedded data from JP-Hide [5],j-steg [16] or [9] Outguess. His detection tool Stegdetect [16] identified over 54,000 images with these detection signatures, but was unable to find the password for any of these images. As such his results were inconclusive.

We believe that the secuity mechanism must be proactive as well as reactive. While it is useful to gain the intelligence about the activities of attackers, the primary goal of the monitor is to provide information security rather than to collect the information. Hence to prevent the malicious communication channel using data fields an exploit specific monitor called Eraser, has been desigtned which will check the payload for the malicious information.

2 Characteristics of a Good Design

The design if any good system must follow particular characteristics. Some of them being:

• Flexibility to support any application-level protocol: The design of the monitor should be independent of any application-level protocol. This can be achieved by separating policy from the mechanism. This separation helps Eraser to support any application level protocol.

Let us consider a simple example in HTTP and SMTP, the message type is indicated in a single word at offset 0. In HTTP, this field contains the request line-method such as "GET" or "POST" when it is an HTTP version it represents a status message type and in the SMTP, this field contains the SMTP commands, such as "MAIL", "RCPT", or "DATA". Some application level protocols such as HTTP allow multiple messages to be received in a single buffer. Therefore, an application level boundary marker is also required. For example for HTTP the message boundary marker is CRLF CRLF and for SMTP, it is CRLF. Different protocols have different message boundries.

• Defensive Design: The design of the monitor itself should not be prone to any attack. In a case when Eraser is spammed with a song, it will have to reconstruct the whole song to check for the malicious content. The system has been designed with the assumption that trusted immutable applications will not contain any hidden information in them. As an example song issued by Sony will not contain any hidden information. The song has been well audited by the Sony for its content. If the case the Sony decides to hide any hidden information, there is very little that Eraser can do. However in the case of mutable applications like MSWord, even though the content has been verified, the data inside the application can still be corrupted. The next subsection discusses about the details of the E-firewall and the authentication protocol mechanism.

3 System Model and Assumption

The proposed system shown in figure 1 consists of a E-firewall which runs on the end hosts and is assumed to be a trusted application and a monitor called as Eraser which runs at Gateway.

3.1 E-Firewall

A conventional firewall relies on the notion of restricted topology and controlled entry points. More precisely it relies on the assumption that every one on one side of the entry point is to be trusted, and that any one on the other side is a potential enemy. A firewall preserves central control of access policy, while reducing or eliminating any dependency on the topology. For the proposed E-firewall we assume that the, insiders are not trusted and can establish the malicious communication channel to send the information outside the network.

For the proposed system a trusted platform will load the E-firewall. The E-firewall will contain the integrity information of the applications. When ever a computer boots up, E-firewall can check for the integrity of the applications with the stored integrity in the firewall or E-firewall can periodically monitor for the integrity of the applications. The storage of integrity is already being used by the current modern day firewall. If some untrusted or mutable trusted application is trying to access the internet the E-firewall will allow the application to send the information. If some immutable trusted application is trying to send the information the E-firewall will put the "identity_stamp" on the trusted application packet. The details about the identity_stamp are explained in the section 3.2. The firewall also maintains a list of dependencies amongst the applications. Besides maintaining a list of trusted applications, the firewall also maintains a list of dependencies amongst the applications. If an application A1 having a high integrity is trying to access the Internet and can write the information to application A2, then the firewall will keep a record of the dependencies. If the integrity of some application A1 is violated then the application A1 might have send some information to the application A2. According to the norms of the trusted computing if the integrity of application A1 is violated then the application A1 should be terminated. However since the application A1 can write to the application A2 or can execute application A3, it might have passed some sensitive information to the application A2 or A3 and hence the application A2 and A3 should also be monitored.

If the integrity of application A1 is violated, then the firewall from the dependency graph will start monitoring all applications to which A1 can write or execute the information. To distinguish a trusted mutable application from an untrusted application, the firewall will put identity_timestamp on the packet.

3.2 Identity time_stamp

The E-firewall will contain the digital certificates of the Eraser in its trusted platform module. (The digital certificate will contain certificate's issuer name, the entity to which the certificate is being issued (aka the subject), the public key of the Eraser and the time stamp). This enables it to establish a SSL connection with the Eraser. Eraser will run an open SSL server. Once a SSL connection has been established Eraser will generate an identity of the distributed firewall along with the hashes to the firewall.

The first 16 bits of the data field are being used for identification and the next 160 bits of the data field are being used for the lamport hash. Since the data field of the trusted immutable appliction is modified, by adding the lamport hash and the identification the checksum is recomputed by adding a one's compliment to the previous checksum.

If the link bandwidth is 10 Mbps and the average packet size is taken to be 350 bytes, the number of distributed hashes required per day by each distributed firewall is $= 24 \times 3600 \times 10 \times 10^6/350 \times 8 = 308571423$ hashes. This is under the assumption that the link is streaming the data at its full capacity. So for n $\times$ 30857142 hashes. This computation is performed by Eraser. These set of hashes help to identify if the packet is coming from trusted immutable applications.

The proposed protocol used for "identity_stamp" by E-firewall is different from distributed firewall in that the IPSec contains Authentication Header(AH), Encapsulated payload (ESP). This header when added to the IP datagram protects the confidentiality,integrity and the authenticity of the data.

The main purpose of the identity_stamp is to separate the data from a immutable trusted application. Since the integrity of the data has already been verified by the vendor and hence it can be assumed that the data does not conatin any hidden information. Hence the proposed protocol is computationally inexpensive. Packet injection and the replay is also not possible, since each trusted immutable packet contains a lamport hash which helps it to identify the trusted application from the untrusted application.

4 The Modules of Eraser

The system as shown in figure 2 comprises the following modules. When the Eraser is invoked, the policy loader loads the policies into specifications. These policies describe the actions to be taken when some malicious content is identified in the data field.

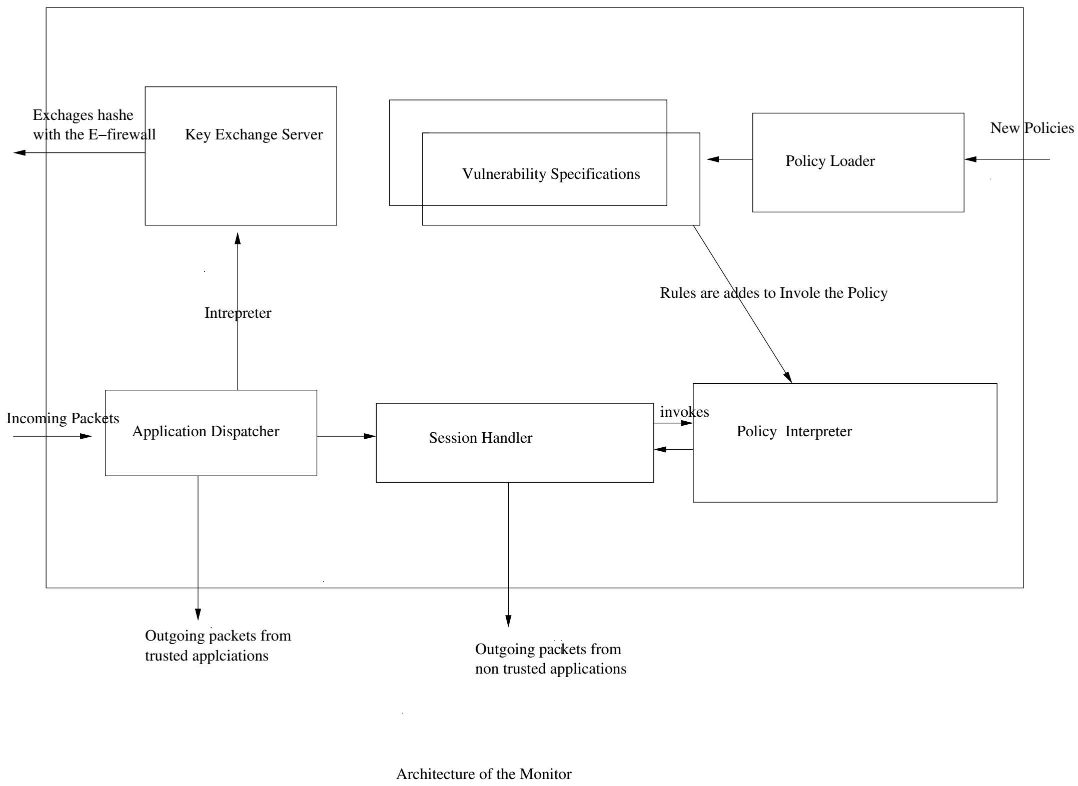


Figure 2: Modules in the Eraser

4.1 Policy Loader: Tasks

This will be invoked when the system starts. Policy loading involves syntax parsing. A syntax tree is also stored in the specifications for inspection purposes. The policy loading will load eraser policies into specifications. The purpose of the policy is to emulate the language stored in the monitor. Since the monitor only checks for the content in the payload, the language of the monitor is type;test;action. Here type specifies the type of the data field, test specifies the particular test which has to be performed to find out any specific vulnerability and the action is the set of actions to be followed when a specific vulnerability is specified. The policy loader also stores the details about the protocol, like the protocol being used the message boundary marker and the actions which have to be performed when a set of data is identified.

4.2 Application Dispatcher

The application dispatcher when initally invoked forks a key exchanger. The identity_timestamp {#id of the distributed firewall, lamport hash} in the packet helps to identify the distributed firewall from which the data is being streamed and the lamport hash helps to identify if the application is trusted or the untrusted application. When the data arrives at the port, the application dispatcher is invoked to determine how the data should be processed. The data used to determine the source of the application. If the data is coming from an immutable trusted application, then the Eraser does not monitor the content in the payload. If the packet is from untrusted application then the payload will be subjected to the rules and policies of the eraser. For this session handler is invoked. Each session is identified by the source port, destination port, source address, destination port, destination address. The source port, source address, destination port, destination address is the hash key which points to the memory location where the payload is parsed for the malicious content.

4.3 Handler

The handler extracts the messages and identifies the type of each message. The content may span in different packets. In such cases the data is reconstructed. Once the data has been reconstructed a call to the policy file is made. The policy file contains the rules and regulations, which identifies the set of actions which has to be taken for processing the malicious content in the payload.

4.4 Policy Interpreter

The policy interpreter invokes the policy file, parses the content of the payload according to the policy file, checks for malicious content in the payload and the data file, and makes a call to the session handler which sends the packet to the outside network.

4.5 Policy language used by the Eraser

The proposed language used by Eraser is less complicated than the Bro Language which is used for scripting the security policies of the Bro network Intrusion detection system. This is because Bro performs network monitoring and intrusion detection for the network layer and above, and also monitors cross-application, cross-session interactions based upon various attack patterns.

The proposed language used by the Eraser is divided into two parts. For the first part the language syntax is type;test;action. The type specifies the type of data in the payload which, has to be reconstructed, test specifies the specific tests to be performed for a particular type when particular attack pattern is detected and the action determines the set of action which has to be taken. The second part of the language specifies generic application level protocol information such as, ports used, the location of the event type, session ID, sequence number or fragment ID in a packet and the message boundary marker. This is independent of the run time condition and is loaded when Eraser starts.

5 Analysis and the Results of the Experiments

The streaming data from the applications in outbound traffic can be classified based on whether the source of the data is trusted or untrusted. If all streaming data is from immutable trusted data then Eraser will not have to check for the malicious content in the payload.

We took the worst case scenario where the streamed data was untrusted data. For the audio around 1/3 of the traffic is TCP and around 2/3 of the traffic is UDP. The UDP packets are of size of 250, 300 and 500 bytes. For audio majority of the data around 60% - 80% is UDP and around 293 and 495 for TCP.

Eraser was running on a 900 Megahertz machine with 512MB of RAM. As shown in the graph 3, the streaming song was collected in a buffer and the song was then parsed for the malicious content in the payload. Each song has a least significant bit encoding. The song was collected in a buffer and the MD5 digest of the LSB was compared

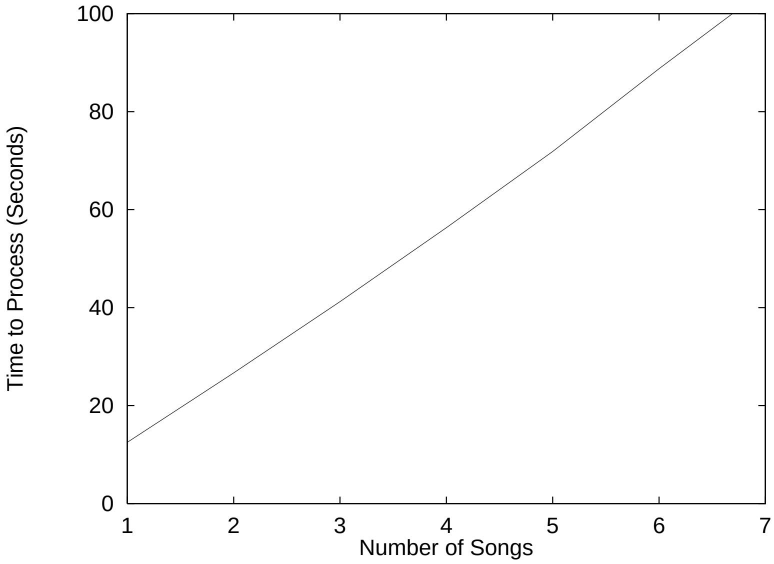


Figure 3: Graph Showing time to process Song having LSB encoding

with the already stored MD5 digest. If there was a match, then the LSB encoding was removed. For 28 songs each of size 100KB, Eraser took, 402 seconds to collect in a buffer, we compute MD5 hash and perform the comparison.

Since the song is having LSB encoding for one bit cost of CPU used is $= 402 * 900 * 10^6 / 2800 * 10^3 = 129214.29 Cycles$

For scalability 50 songs, each of size 100 KB were again streamed at different bandwidths. As shown the graph in the figure 4 the bandwidth increases, time to process the songs also increases and since Eraser is limited in its capability by the processing speed and the capability of the memory, it will start to drop the packets if the incoming data is beyond its capability. The point where Eraser starts to drop the packets is where the scalability of the system can be calculated.

For 127 KB/sec, out of 50 songs 40 songs were dropped. This gave a drop rate of around 80%, for 75 KB/sec drop rate was around 60%. From the graph it can be seen as the link bandwidth increases drop rate increases. Even for 10 KB/sec, drop rate was around 30%.

The increase in drop rate can be attributed to two factors. We believe that the primary reason being that the system is implemented in JAVA and second one being that the Eraser is running on a 900 Megahertz Computer. With better processing power better results can be achieved.

6 Discussion

From the experiments it can be seen that as the number of songs increases the time to process the packets increases linearly. The result is quite obvious because in order to remove the LSB encoding in the song we are using MD5 hash digest and performing linear comparison.

The time to process the packet will again depend upon the test which has to be performed and the action which has to be taken by the Eraser to remove the content. The policy language used by the eraser is type;test;action. If the exploit is known then the suitable tests can be performed to remove the malicious communication channel. Malicious content removal by Eraser is 100% however CPU overhead to remove is dependent upon the tests and the actions.

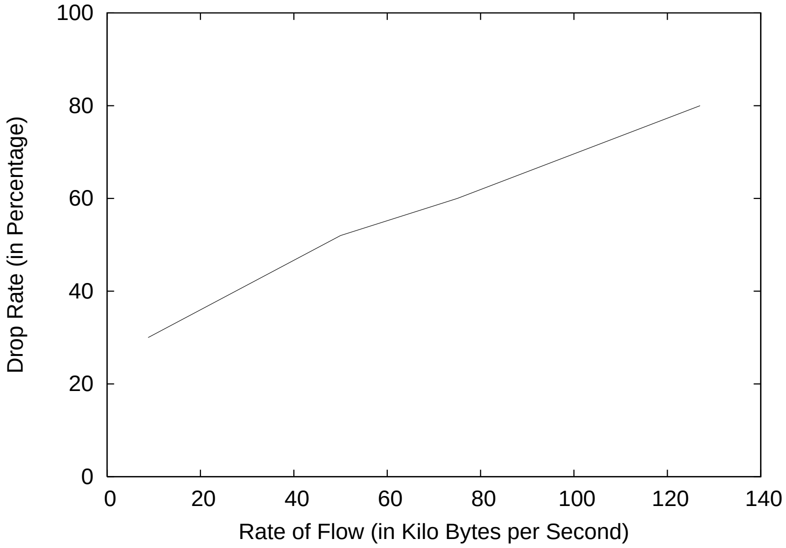


Figure 4: Graph Showing Scalability of the Eraser

So as the number of songs increases the comparison time increases. For LSB encoding and linear comparison took around 402 seconds for 28 songs. This is quite a significant amount of unacceptable delay. This futher motivates our design of using identity_stamp for classifying the packets from trusted mutable applications and immutable applications. If the packets are from trusted immutable applications then the Eraser will not have to check for the malicious content in the payload and hence end to end latency due to the monitoring of payload can be avoided.

Scalability and cost to remove the malicious covert information is dependent upon the test which has to be performed and the action which has to be taken to prevent it. If the particular tests and exploits are not known then Eraser will not work. Also if an adversary is successful in providing incorrect the content information Eraser will fail to remove the malicious content in the payload. Another possible attack can be the compromise of the application streaming trusted immutable data.

7 Research Contributions and the Future work

We have presented here the design architecture of Eraser: An exploit specific monitor which checks for the malicious content in the payload. We have also described the design and implementation of E-firewalls. We have presented the classification of trusted applications into mutable trusted applications and immutable trusted applications. Immutable trusted applications provide the inherent advantage that the integrity is verified, by the vendor and hence may be assumed to have no hidden information.

• E-firewall stores the dependencies amongst the applications. This is an incremental advantage over the existing firewalls. Currently the dependencies are generated manually amongst the programs. In the proposed system these dependencies help to decide the flow of information amongst the applications. If the integrity of some application is compromised then from the dependency graph, E-firewall and the eraser will know what other application needs to be monitored. E-firewall can be used not only to prevent the malicious flow of information but also to stop the applications from where it is receiving information. Currently dependencies are generated manually ongoing work is development of algorithms which can be used for the generation of dependencies.

• We have described here the idea of identity_stamp, a light weight authentication protocol as used by the distributed firewalls. The packets from the trusted immutable applications contain the identity of the distributed firewall and the lamport hash. This identity and the lamport hash help to identify the packets are from a trusted immutable application. E-firewall puts the identity_timestamp on the mutable applications.

• We have presented the classification of the trusted applications into two parts. First classification is the trusted mutable applications and the second is immutable trusted applications. Immutable trusted applications provide an inherent advantage over mutable trusted applications since data has been verified by the vendor and will not contain any malicious information. However this is only under the condition that the vendor does not decide to hide the malicious information.

• Eraser has been designed to prevent the insider threat problem. The main objective of Eraser is to prevent a malicious communication channels using the data field in outbound traffic. Eraser can also be used to enforce site specific policies. The proposed language used by the eraser is type;test;action. The proposed language provides the inherent advantage over the other languages in the term that the language can be used to enforce the site specific policies and hence it can act as a filter for common occurring words.

• The deployment issue can also be a focus of further research. Eraser can either be built in a special purpose box or it can be integrated into an active monitoring system like a firewall or IPS which monitors outbound traffic.

References

[1] *Eliminating Steganography in Internet Traffic with Active Wardens.* 5th International Information Hiding Workshop on Information Hiding, 2002.

[2] Steven M. Bellovin. Distributed firewalls. In *login*, 1999.

[3] G.J.Simmons. "the prisoner's problem and the subliminal channel". In *"Advances in Cryptography: Proceedings of Crypto-83"*, 1984.

[4] A. Havill. The spy who staed out in the cold: The secret life of double agent robert hanssen. In *St. Martin's Press*, 2001.

[5] JP Hide and Seek. http://linux01.gwdg.de/alatham/stego.html.

[6] Miroslav Goljan Jessica Fridrich and Rui Du. Shared resource matrix methodology: An approach to identifying and usage and timings channel. In *ACM Transaction on Computer Systems*, 1983.

[7] N.F.Johnson and S. Jajodia. Stegnalysis : The investigating of hidden information. In *Proceedings of IEEE Information technology Conference*, September 1998.

[8] N.F.Johnson and S.Jajodia. Exploring the steganograpy:seeing the unseen. In *ACM SIGCOMM' 04*, 2004.

[9] Outguess. http://www.outguess.com.

[10] Vern Paxson. Bro: A system for detecting network intruder in real time.

[11] Umesh Shankar and Vern Paxson. Active mapping: Resisting nids evasion without altering traffic. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 376–382, May 2003.

[12] Daniel Sieberg. Bin laden exploits technology to suite his technology needs. In *http://www.sscnet.ucla.edu/geog/gessler/topics/crypto-bin-laden.htm*, 2002.

[13] Abhishek Singh, Ola Nordstrom, Chenguhai Lu, and Andre L M dos Santos. Malicious icmp tunneling defense against the vulnerability. In *The Eight Australian Conference on Information Security and Privacy*, 2003.

[14] Abhishek Singh, Ola Nordstrom, Chenguhai Lu, and Andre L M dos Santos. Using consistency check to prevent the malicious tunnels. In *CNIS*, 2003.

[15] Brenda Stardom. Terriorist use of steganography. In *http://www.brendastardom.com/arch.asp?ArchID=48*, 2002.

[16] stegdetect. http://www.stegdetect.com.