A Trust-based Model for Collaborative Intrusion Response

Kapil Singh Norman C. Hutchinson

Department of Computer Science,
University of British Columbia,
Vancouver, Canada
E-mail: {singh,norm}@cs.ubc.ca

Abstract

Intrusion detection systems (IDS) are quickly becoming a standard component of a network security infrastructure. Most IDS developed to date emphasize detection; response is mainly concentrated on blocking a part of the network after an intrusion has been detected. This mechanism can help in temporarily stopping the intrusion, but such a limited response means that attacking is free for the attacker. The idea behind our approach is to frustrate the intruder by attacking back. This requires developing a sense of trust in the network for the attacked host and establishing proof of the attack so the attack-back action can be justified.

To develop this trust model, we propose a protocol that allows the attacked host to prove to the attacker's edge router that it has been attacked. The model is quite flexible, and based on the level of trust developed for the host, an appropriate countermeasure is taken. Besides attack-back, other possible responses could be blocking a part of the network and use of network puzzles to limit the attacker's access to network resources.

We believe that the attack-back approach would certainly demoralize novice attackers, and even expert attackers will think twice before attacking again. In addition, the protocol prevents a host from pretending that it has been attacked. We are building a system that can handle a majority of known attacks (signature-based). We are also exploring the idea of adding a third trusted party into the system in order to provide countermeasure action for novel attacks (anomaly-based).

1 Introduction

With the rapid expansion of computer networks during the past few years, these are fast becoming a subject to a variety of intrusions. The amount of financial losses due to cyber attacks have grown manyfold over the years. It is estimated that the worldwide impact of malicious code was $13.2 billion in the year 2001 alone [10]. The constant increase of attacks against networks and their resources causes a necessity to protect these valuable assets. In fact, security is a constant race between the attackers and the defenders. The attackers are getting smarter day by day and hence the need for smarter security solutions always remains.

An effective solution to the network security problem requires a collaborative effort from various entities in the Internet, such as routers, gateways and end hosts. With the idea of collaboration comes the need for trust among the cooperating parties. The open nature of the Internet makes it difficult to trust anyone in the cyber world and this is the motivation behind our work. This paper describes a model to develop trust among various network elements, effectively enabling them to take a collaborative action against network attacks.

Intrusion detection systems (IDS) are quickly becoming a standard component of a network security infrastructure. IDS are the last line of defense against computer attacks behind firewalls, secure architecture design and penetration audits. Most IDS developed to date emphasize detection; response is mainly concentrated on blocking a part of the network after an intrusion has been detected. This mechanism can help in temporarily stopping the intrusion, but such a limited response means that attacking is free for the attacker. The idea behind our approach is to frustrate the intruder by attacking back. Our trust model provides the required trusted environment for taking appropriate countermeasures and also justifies the extreme measure of attacking back.

The remainder of this paper is structured as follows. In Section 2 we describe the protocol for trust generation between various network elements. Section 3 presents the attack-back approach and an algorithm for selecting the appropriate response. Section 4 describes the implementation of the prototype and analysis of our experiments. We describe limitations and extensions in Section 5. Section 6 reviews the related work followed by conclusions in Section 7.

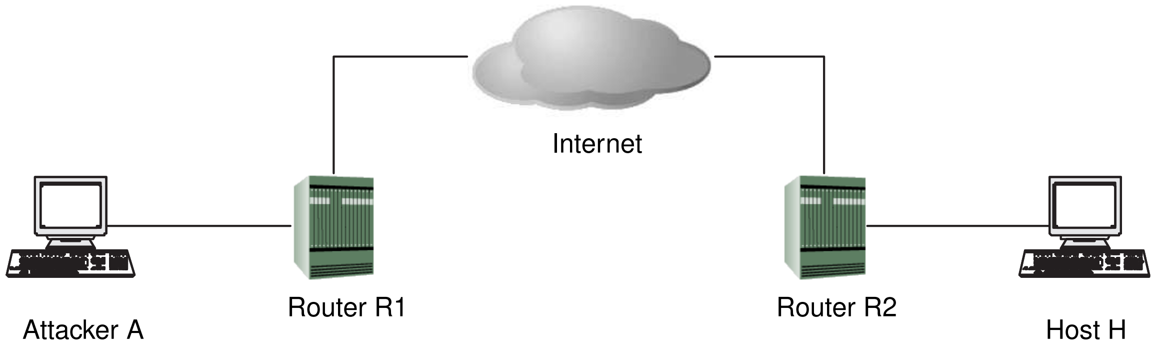

Figure 1: Typical network architecture.

2 Protocol Description

In this section, we describe the protocol for developing trust among various network entities. This trust is needed to make sure that the ability to take proper action is not abused by a malicious node to disrupt other nodes' communications.

2.1 Overview

The IDS at the host machine observes the incoming network traffic for signs of possible intrusions. Once the IDS is suspicious that the host is being attacked, it informs the edge routers to start monitoring this traffic chain. When the IDS is convinced that the host is under attack, it sends the intrusion-specific information in an encrypted form to the attacker's edge router. The router, in turn, compares its own logs against the received information to decide on the appropriate response. The logs of the victim's edge router can also be used to improve this trust level.

2.2 Assumptions

We assume that there would be some anti-spoofing mechanism in place, which limits source address spoofing. A lot of research is being done in this area [5][13] and we are confident that a deployable anti-spoofing mechanism would be available for the Internet in near future. Also, the routers would support logging of traffic and provide additional functionality offline. Our mechanism motivates edge router of the attacker to contribute, else it risks loosing its connectivity to the host.

2.3 Protocol for developing trust

Figure 1 shows a typical system. 'A' is the attacker machine having the edge router 'R1'. 'H' is the host machine having the edge router 'R2'. IDS is installed at the routers and the host machine.

The steps involved in this protocol are:

Step 1. The IDS at H scans for any intrusion attempt at the network interface. If the initial state of an intrusion is detected, the process moves to the next step. An example of the initial state is the receipt of port scan packets from a different subnet. In the case of denial-of-service attacks (such as the *ping of death*), this initial state is the first attack packet. The initial state is just a suspicion state and is not a confirmation of the attack.

Step 2. H sends a *Start-Logging* message to inform the routers to start monitoring the traffic from A, specifying its own IP Address, IP_H, and the IP Address of A, IP_A. In case of intrusions that are interactive and bidirectional, such as the attacks aiming to gain unauthorized access, this is done using the Watermarking technique [11].

$$H \rightarrow R1 \;:\; IP_H, IP_A, \textit{Start-Logging} \tag{1}$$

After receiving this message, the router starts logging the traffic coming from A. This logging is time controlled and if the router doesn't receive any confirmation message from the victim within a timeout period, it stops logging. This prevents overloading of the router when there is a false suspicious state and hence no confirmation is received from the host (*Step 3*). An alternative to the time-bound garbage collection could have been an extra *Stop-Logging* message from H to the routers, but this can be exploited by intentionally sending an initial *Start-Logging* message and no confirmation. The routers would keep logging indefinitely waiting for confirmation from the host.

Step 3. When the IDS at H is convinced that it is an attack, it sends the *Verify-Attack* message to A's edge router R1, specifying the attack signature (*AttackScenario*) and any parameters associated with that attack.

$$H \rightarrow R1 \;:\; IP_H, IP_A, \textit{Verify-Attack}, \\ \textit{AttackScenario}, \textit{parameters} \tag{2}$$

Step 4. After receiving the *Verify-Attack* message from H, R1 verifies the attack claim offline by running the IDS on its own network logs. Doing this processing offline does not impact the general functionality of the router. The IDS just looks for the attack signature specified by the *AttackScenario* and its parameters and hence the processing is much faster than if it were scanning for all possible attacks.

Step 5. Based on the level of trust developed, an appropriate action is taken by the router. The possible responses are described in Section 3.

Step 6. If the attacker router R1 doesn't respond to H's request, the request is sent to the host's edge router R2 (or some intermediate router that sees the majority of traffic between A and H) and *Steps 3-5* are repeated.

2.4 Encryption Algorithm

In order to prevent eavesdropping on the control messages, the messages are encrypted. This ensures that the messages are not modified in the network path and are not read by the attacker. Also, the identity of the victim is certified by the use of digital signatures.

M	The starting message
h_M	A SHA1 hash of M
e_M	Message M encrypted with AES
K^{AES}	AES key
K_H	Private key of host H (RSA)
K_H^+	Public key of host H (RSA)
K_{R1}	Private key of router R1 (RSA)
K_{R1}^+	Public key of router R1 (RSA)

Table 1: List of Variables.

Table 1 lists the variables used to describe the algorithm that follows. The certifiable public keys of the routers and the host are well-advertised (e.g., using web pages, e-mail, or IRC). We next describe the processing of the messages at the sender (encryption) and the receiver (decryption).

At Host H (before sending)

1. h_M is generated from the message M.
2. Create a block of data "h_M:M".
3. Encrypt the block using K^{AES}. We used K^{AES} to encrypt the block instead of K_{R1}^+ for two reasons. Firstly, the attacker can never guess the key as it can be changed for every new message. Secondly, encryption using AES key is faster and since the message block forms a major chunk of data being sent, encryption using K^{AES} results in better performance.
4. Encrypt the K^{AES} with K_{R1}^+ (to make it secret).
5. Sign the h_M with K_H to certify the identity of the sender H.
6. Create a block of data "signed data:encrypted AES key:encrypted block".

This data is sent to the router R1.

At Router R1 (on receipt)

1. Unpack the block to get "signed data", "encrypted AES key" and "encrypted block".
2. Decrypt the encrypted AES key using K_{R1} to get K^{AES}.
3. Decrypt the encrypted block using K^{AES} and split it to get h_M and M.
4. Create a SHA1 hash h'_M of the message M.
5. Compare h'_M with h_M to make sure that the message is received in its original form and not changed by a man-in-the-middle attack.
6. Verify the signed hash with h'_M, using K_H^+. This certifies the identity of the sender and prevents the attacker from masquerading as the host.

All messages from the victim to the edge routers are encrypted using this algorithm before sending (R1 being replaced with the appropriate router, if needed). Even after use of encryption, there is still possibility of a message replay attack. Our protocol prevents this by discarding all logs after an action has been taken against the attacker. The old information in the replayed message does not match the new logs and is hence discarded.

3 Responses

Depending on the extent a router is convinced about an attack, it takes appropriate action ranging from no action in case of a failed proof to extreme measure of an attack-back. For our system, we have considered network puzzles [12], blocking a part of the network and attack-back as possible responses, but the framework is flexible enough to add other response types.

3.1 Heuristics for response selection

An appropriate response for an intrusion attempt is determined by two conditions – firstly, how much do you trust the host who is claiming to be a victim and secondly, how much are you suspicious of the "so-alleged" attacker. We use two parameters to form the heuristics, *Level of Trust* and *Suspicion Rank*, which correspond to the two conditions. Level of Trust is determined by

```
TakeAction(A, H, Scenario)
  if (canBeProved(Scenario))
    if (proveAttack(A, H))
      LoT(H)++, SR(A)++
      SelectResponse(LoT(H), SR(A))
    else
      LoT(H)--
  else
    SelectResponse(LoT(H), SR(A))

SelectResponse(LoT, SR)
  trust:
      1 : [LoT ≥ LoT_max]
      0 : [LoT_min < LoT < LoT_max]
      -1 : [LoT ≤ LoT_min]
  suspicion:
      1 : [SR ≥ SR_max]
      0 : [SR_min < SR < SR_max]
      -1 : [SR ≤ SR_min]

  responseValue = trust + suspicion

  if(responseValue ≥ 2)
    Attack-back(A)
  else if (responseValue == 1)
    BlockAllTraffic(A)
  else if (responseValue == 0)
    BlockTrafficToVictim(A, H)
  else if (responseValue == -1)
    UseIPPuzzles(A)
  else if (responseValue ≤ -2)
    NoAction(A)
```

Figure 2: Pseudo code for response selection.

the extent to which the router is convinced from its own logs about the attack claimed by the victim. This value is incremented if the attack is successfully proved and reduced if the proof fails. The reduction in the Level of Trust on a failed proof would discourage false attack claims, possibly from an attacker masquerading as a victim. Suspicion Rank maintains a count of the intrusion attempts made by a particular attacker. It is high for an attacker with a higher number of attempts that are successfully proven as attacks. The values of the thresholds for Level of Trust (LoT_{max} and LoT_{min}) and Suspicion Rank (SR_{max} and SR_{min}) are determined by the level of sensitivity acceptable to the router. For example, if the router wants to be "soft" in its responses, it can choose large values for LoT_{max} and LoT_{min} and large values for SR_{max} and SR_{min}.

As described in Section 2.3, logging at the routers can only start after the suspicion state. So, in order to prove an attack, we effectively need more than one attack packet. There might be attacks that consist of just one packet, for example, a single-packet buffer overflow attack. Our system is able to handle such "non-provable" attacks based on existing values of the aforementioned parameters. Effectively, there is no change in the value of LoT and SR, but still these values can be used to decide the response. In real life, this is analogous to having more suspicion on a criminal which has a notorious criminal record. The punishment for such a criminal keeps increasing with each proved crime, which is similar to having a stronger response for each proved attack in our algorithm. The complete algorithm for selecting the appropriate response is given in Figure 2. The idea is simple - if the attacker has a bad track record of being notorious and the victim has a good record of being truthful, then the response is harsher against the attacker. In order to be fair for a known attacker who wants to redeem himself, the value of Suspicion Rank is decremented based on time.

3.2 Attack-back Approach

The best defense is a good offense.

This well-known proverb is also the motivation for our idea. Blocking a part of the network can temporarily stop the intrusion, but the attacker doesn't loose anything. The idea behind our approach is to frustrate the intruder by attacking back. This approach will surely demoralize the novice attackers and even the expert attackers will think twice before attacking again. Since majority of the Internet attacks are generated by novice attackers, the attack-back approach can be quite effective. We also understand that this approach can be exploited by attackers posing as victims to launch attacks on innocent hosts. But our protocol strikes down this possibility. The protocol is fool-proof as the routers don't need to have a prior belief in any host and their action is solely based on their own view of the traffic. It also removes the possibility of a wrong assessment by the router because the attack has been acknowledged at more than one place, i.e., the victim and the edge router(s).

4 Prototype

We have developed a prototype of our automated intrusion response system. We tested the prototype against some well-known attacks and were successful in getting the desired results. We were successful in generating the appropriate response as per our response heuristic described earlier. The main objective of our prototype was to prove our concept of trust generation between the network entities and automate the selection of appropriate responses. We used NetSTAT [9] as the IDS for our prototype, but any state-based IDS would work for the system.

4.1 NetSTAT

NetSTAT [9] is a network-based IDS developed by the Reliable Security Group at UCSB. It is based on the state transition analysis technique (STAT) [6]. The NetSTAT approach models network attacks as state transition diagrams, where states and transitions are characterized in a networked environment. There were several incentives to choose NetSTAT as the IDS for our system. Firstly, NetSTAT is state-based and hence it is easier to differentiate between the suspicious state and the conviction state. Secondly, NetSTAT allows offline detection of attacks from network logs. This is required by our system, since the edge routers work on the network logs offline to evaluate the proof of the attack. Thirdly, NetSTAT uses an intrusion detection language STATL [4] and hence the responses are easier to program. We just made minor changes to the STATL scenarios and responses without touching the NetSTAT core.

4.2 Experience with the prototype

We experimented with several well-known attacks by generating the attacks in our testbed. Though the vulnerabilities exploited by these attacks were fixed in the Linux version we used, the attacks could still be detected. Our system is limited by the performance of the IDS for detecting the attacks and we assume that the IDS would update itself periodically with the signatures of the recently known attacks. Due to lack of space, we are only describing one attack scenario here, but still one scenario is sufficient to prove our concept as there is no change in the protocol between different scenarios.

4.2.1 Ping of Death attack

IP packets can be up to 65535 octets long, which includes the header length (typically 20 octets if no IP options are specified). Packets that are bigger than the maximum size the underlying layer can handle (the MTU) are fragmented into smaller packets, which are then reassembled by the receiver. For Ethernet style devices, the MTU is typically 1500. An ICMP ECHO request "lives" inside the IP packet, consisting of eight octets of ICMP header information followed by the number of data octets in the "ping" request. Hence the maximum allowable size of the data area is 65535 - 20 - 8 = 65507 octets. It is possible to send an illegal echo packet with more than 65507 octets of data due to the way the fragmentation is performed. The fragmentation relies on an offset value in each fragment to determine where the individual fragment goes upon reassembly. Thus on the last fragment, it is possible to combine a valid offset with a suitable fragment size such that (offset + size) $>$ 65535. Since typical machines don't process the packet until they have all fragments and have tried to reassemble it, it is possible to overflow the 16 bit internal variables, which can lead to system crashes, reboots, kernel dumps and the like.

NetSTAT successfully detected the ping of death attack packets. On receipt of the first such packet, the host informed the edge routers to start logging and after some threshold number of packets were received, it informed the routers to take appropriate action passing them the scenario as ping of death. The attacker's edge router first blocked the traffic of the attacker for a period of time and since the attack did not cease, the router finally attacked back.

5 Related Work

Katerina et el. [1] describes a mechanism in which a victim can request the relay closest to the attack source to block traffic from the attack source. The mechanism uses a filter management scheme called AITF [2] to guarantee secure communication between the victim and the relay (the attacker's edge router). AITF uses a three-way handshake to verify that the request is real and prevents eavesdropping by a malicious node. But it fails if the malicious node is located on the communication path. This problem does not exist in our system because the encryption algorithm ensures that the message is read only by the node it is intended for. Also, AITF assumes that the claim made by the victim is always true and if the attacker's edge router doesn't comply to the request, it loses the connectivity to the victim. Thus, AITF is vulnerable to fraudulent attack claims. In our protocol, the edge routers prevent this by verifying the claim locally using its own view of the network logs.

To the best of our knowledge, we are the first to use the *proof-based* attack-back approach as a response to an attack. There are numerous examples of attacking back in the past [8][3] but none of them provide a proof to support the response. Hence the individuals or companies involved in such action have been widely criticized. One such recent incident happened in December 2004 when Lycos Europe released a screensaver "Make Love not Spam" [3] that bombarded spam websites with data to try to increase the cost of running such sites. The end result was that some spam sites were completely overwhelmed by the traffic directed their way and Lycos was criticized for encouraging vigilantism. The advantage our system has over their approach is the development of trust before attacking back. In our system, the edge routers are acting as "witness" to the attack against the victim and hence provide a proof supporting the attack-back action.

6 Limitations and Extensions

While we have been successful with our prototype, we recognize a number of limitations and potential challenges facing those building a complete automated intrusion response system. In this section we discuss these and discuss the extensions we plan to add as part of future work.

6.1 Limitations

The biggest dependency of our system is on the efficiency of the underlying IDS. We haven't developed any new IDS for our system but instead we have concentrated on the trust-based response action after the IDS has detected the intrusion. Another limitation is that the system is effective only in case of attacks detected by network-based IDS. It is not easy for the router to locally reproduce the conditions prevalent at a particular host in order to determine the validity of the attack claim. But this can be regarded as a special case of a "non-provable" attack and can be handled by our system as described in Section 3.1.

6.2 Anomaly-based Response

Our prototype can successfully handle all known attacks whose signatures are available to the IDS. As part of a future extension, we plan to use the same concept for novel attacks whose signatures are not yet known. We have explored this possibility by addition of a universally trusted third party. This party could be an organization or a set of hosts using some machine learning approaches to detect any abnormal network flows [7].

In this case, the victim would send its logs to the trusted host. The trusted host would determine if the given claim is a valid attack and then generate a token for the attacker's edge router. The edge router would take appropriate action based on this token. If the trusted party is not fully convinced by the victim's claim, it can ask the edge router to send its logs concerning the particular victim. The AI component of the IDS and the trusted party is currently being investigated.

7 Conclusions

In this paper, we presented a trust-based model for an automated response to network intrusions. If network elements can collaborate, then they can be more effective against intrusions. We have developed a protocol to create trust among various network elements, so that they can coordinate to take appropriate action against the attack. The protocol is fool-proof against false attack claims and prevents an attacker, posing as a victim, from using the protocol to generate attacks against innocent hosts. The encryption algorithm prevents any eavesdropping on the protocol messages being passed in the network.

This is one of the first attempts to use an "attack-back" approach as a response for an attack. If this approach is harnessed in a controlled manner, it can be an effective weapon against the intruders. Our protocol provides this control by using the proof-based approach. We have also defined an algorithm for selecting the appropriate response based on the level of trust developed for the victim. The framework is quite flexible allowing for any new response type to be added.

References

[1] K. Argyraki and D. R. Cheriton. Loose Source Routing as a Mechanism for Traffic Policies. In *Proc. of FDNA'04: ACM SIGCOMM Workshop*, Portland, USA, 2004.

[2] K. Argyraki and D. R. Cheriton. Protecting Public-Access Sites Against DDoS Attacks. Technical report, Stanford University, May 2004.

[3] BBC News World Edition. Anti-spam plan overwhelms sites, December 2004. http://news.bbc.co.uk/2/hi/technology/4061375.stm.

[4] S. Eckmann, G. Vigna, and R. Kemmerer. STATL: An Attack Language for State-based Intrusion Detection. *Journal of Computer Security*, 10(1/2), 2002.

[5] I. Hamadeh and G. Kesidis. Packet Marking for Traceback of Illegal Content Distribution. In *Proc. of International Conference on Cross-Media Service Delivery (CMSD)*, Santorini, Greece, May 2003.

[6] K. Ilgun, R. Kemmerer, and P. Porras. State Transition Analysis: A Rule-Based Intrusion Detection System. *IEEE Transactions on Software Engineering*, 21(3), March 1995.

[7] M. V. Mahoney. Network Traffic Anomaly Detection Based on Packet Bytes. In *Proc. of SAC'03*, Melbourne, Florida, 2003.

[8] D. Radcliff. Can You Hack Back? *NetworkWorld*, June 2000.

[9] G. Vigna and R. Kemmerer. NetSTAT: A Network-based Intrusion Detection Approach. In *Proc. of ACSAC'98*, Scottsdale, Arizona, December 1998.

[10] B. Waite. Malicious Code Attacks Had $13.2 Billion Economic Impact in 2001. *Computer Economics*, January 2002.

[11] X. Wang, D. S. Reeves, S. F. Wu, and J. Yuill. Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework. In *Proc. of the 16th International Conference on Information Security: Trusted Information*, 2001.

[12] Wu-chang Feng. The case for TCP/IP puzzles. In *Proc. of FDNA'03: ACM SIGCOMM Workshop*, Karlsruhe, Germany, 2003.

[13] A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, USA, May 2003.