Ruofan D.
Peerapat L.
CS4404 - Final Project Report

Active Internet Traffic Filtering (AITF) Protocol

* Abstract

This paper will discuss in detail how to design and write a program to incorporate the Active Internet Traffic Filtering protocol into a network system to combat denial-of-service attacks. By laying bare the individual components characteristic of an effective implementation of the AITF protocol, and presenting a simulation such an implementation, empirical data will be used to support the claims put forwards by this paper in regards to the AITF protocol. It will also be shown that even in the event of members of the network not adhering to the AITF protocol, the performance of the network as a whole will always perform at the very least at the same level as it would have, without AITF implemented.

0 Introduction

The primary objective of DDoS attacks is to make a network resource unavailable. Though there are several types of DDoS attacks that may be executed, the primary attack that will be used in this simulation will be the *UDP flooding attack*. Conventional methods of combatting DoS attacks involve the gateway routers of the victim installing a filter for each venue of attack directed at the victim. With the possibility of a botnet being the perpetrator, the limited supply of filters at the victim's gateway router will be rapidly exhausted. By distributing the responsibility of filtering out malicious traffic to the gateway routers closest to the sources of the attacks, the bottleneck to filtering out the traffic caused by the victim's gateway is dealt with. This method of distributing the load of traffic filtering so that each gateway router closest to an attack source contributes is called *Active Internet Traffic Filtering* (*AITF*). This paper will explore how to implement *AITF*, and the benefits to a network's security by doing so.

1 Design of Implementation

1.2 Overview

The AITF protocol combats DDoS attacks by eliminating them as close to the source as possible. It is able to accomplish this by inserting route records into a *RR shim layer* in each packet. The victim host and the victim's gateway router will obtain information about the packet's route through the packet's RR shim, and communicate with the router most closest to the attacker to have it filter out the undesired traffic. This holds an advantage over conventional local filtering at the victim's gateway as the quantity of filters that must be used to thwart an attack is able to be distributed amongst routers near the sources.

Another key feature of the AITF protocol that allows it to prevent DDoS attacks is *escalation*. Escalation is the AITF protocol's means of adapting to spoofed data and uncooperative gateway routers. When escalation is performed, the AITF protocol will be carried out between the victim's gateway router and the next closest router after the attacker's gateway router in the *flow* between the victim and the attacker, rather than the attacker's gateway router. A flow is defined as: the set of all packets that share a common *recorded path* suffix. A recorded path is defined to be the sequence of IP addresses that correspond to: the packet's source, the list of routers specified in the RR shim, and the packet's destination.

Whether to perform escalation is decided by the *policy module*. The policy module is a set of predetermined criteria that will be tailored to the desired level of performance of the protocol. Other issues addressed by the policy module, for instance, include when to start executing AITF, and for how long undesired flows should remain filtered for.

1.2.1 Assumptions

The simulation of the AITF protocol will be performed under certain assumptions:

1. The victim supports AITF.
2. The victim's gateway router supports AITF and is non-malicious.
3. The DDoS attack being performed is the UDP flooding attack.
4. The rate at which a filter request is sent out by a victim end-host is once per second.
5. The attacker is not ingenious to perform IP spoofing attacks.

1.3 Threat Model

The following threat model of the AITF protocol will address the potential threats that the AITF protocol seeks to stop, and key components that will be used by the AITF protocol to do so.

1.3.1 Threats

The primary threat that the AITF protocol seeks to address is the UDP flooding DDoS attack. The UDP flooding attack will be directed by an attacker end-host at a victim end-host. The victim end-host will use the AITF protocol to protect its *goodput*, or good throughput, and prevent its resources from being expended on undesired traffic. The attacker may attempt to thwart the AITF protocol through the following modification of its UDP datagrams: *IP spoofing, RR spoofing*. The attacker may perform IP spoofing by spoofing its IP address such that it seems like another end-host or router is sending its traffic. The attacker may also perform RR spoofing in conjunction with IP spoofing in an attempt to impersonate a router. IP spoofing, however, will not be tested in this implementation due to time constraints. The attacker may attempt to thwart the AITF protocol by behaving differently: *Lying*. When the attacker lies, it will stop its DDoS attack while its gateway router's temporary filter is up, and resume once the temporary filter has been removed. This is particularly effective only when there are compromised routers along the undesired flow.

1.3.2 Key security features

<u>Netfilter</u>

A *netfilter* will be used by all border gateway routers as part of the AITF protocol. Gateway routers will use their netfilters to establish *filter rules*, which will enable undesirable incoming traffic to be dropped. They may also use their netfilters to detect UDP datagrams sent to end-hosts within their subnets, as is required in the AITF protocol's *4-way handshake* (refer to *Figure 1*).

Additionally, AITF-enabled end-hosts will use their netfilters to analyze incoming packets based on their *policy module*, and remove the RR shim of incoming AITF packets before forwarding them to the application layer. Netfilters will also be used to insert the RR shim by gateways.

Filter Rule

A *filter rule* is a rule that is installed by a router's netfilter to prevent packets of an undesired flow from being forwarded. A filter rule primarily identifies a packet by its source and destination IP addresses.

Protocol 253

The protocol field in the IP header will be modified to indicate whether or not the packet has been modified by the AITF protocol to contain an RR shim. The protocol: 253, will be used to indicate the presence of the RR shim, as it is reserved for "experimentation and testing".

RR Shim

For the victim's gateway router to be able to identify undesirable flows and to perform escalation, it needs to be able to determine a packet's route. This information will be stored in packets through the use of a *shim header*. A shim header is a manually inserted header that allows additional information about the packet to be recorded. The route record shim (*RR shim*) used in this implementation will be a 56 byte shim that will serve to record the following information: the original protocol of the packet prior to the insertion of the RR shim; a hop count counter; and six pairs of router IP addresses and their respective R values (refer to *Figure 0*). It should be noted that only AITF-enabled actors will be capable of modifying the RR shim.

Figure 0: The RR Shim

```
Bytes     0   1   2   3   4   5   6   7
          +---------------+---------------+
...       |Network Layer                  |
          +---------------+---------------+
          |Protocol       |Counter        |
          +---------------+---------------+
          |IP_Addr_0      |32bit_R_0      |
          +---------------+---------------+
          |IP_Addr_1      |32bit_R_1      |
          +---------------+---------------+
          |IP_Addr_2      |32bit_R_2      |
          +---------------+---------------+
          |IP_Addr_3      |32bit_R_3      |
          +---------------+---------------+
          |IP_Addr_4      |32bit_R_4      |
          +---------------+---------------+
          |IP_Addr_5      |32bit_R_5      |
          +---------------+---------------+
...       |Transport Layer                |
          +---------------+---------------+
```

R Value

To prevent the attacker from forging a path by spoofing router IP addresses in the RR shim, each AITF enabled router will insert a *random R value* along with its IP address into the shim layer of each packet it receives. The R value is calculated via the following equation: $R = hash_{key}(D)$, where *key* is a private key of a router, *hash* is a keyed hash function, and *D* is the packet's destination IP address. It should be noted that the R value used in our simulation is 32 bits. Cooperative AITF-enabled routers will reject filter requests with incorrect R values, and notify the requester that it is RR spoofing.

Nonce1

The attacker's gateway router will need verification that the router that is sending a filter request to it is actually the victim's gateway router. To accomplish this, the attacker's gateway router will send UDP datagrams containing the undesired flow, *F* and a *nonce*, to the *victim*. A nonce is a randomly generated 32 bit value. The actual victim's gateway router should be able to intercept the packet and send it back to the attacker’s gateway. This determines the authenticity of the victim’s gateway router.

<u>Nonce2</u>

The victim’s gateway router will need verification that the router that is replying to its filter requests is indeed the attacker’s gateway router. To accomplish this, the victim’s gateway router will insert a second nonce along with the first nonce it intercepted in its reply to the attacker’s gateway router. The second nonce will be randomly generated by the victim’s gateway.

<u>Filter Table</u>

Each router stores its filter rules in a *filter table*. Filter rules that are entered into the filter table will remain there for up to T_{tmp} seconds, or until they are removed by the router. These filter rules are considered to be temporary filter rules. Once a filter rule is removed from a router’s filter table, it is entered into the router’s *shadow filter table*.

<u>Shadow Filter Table</u>

Filter rules that are entered into the shadow filter table will remain there for up to $T_{long} >> T_{tmp}$ minutes. Before installing a filter rule, the router will check if the filter rule already exists in the shadow filter table. If it does, then the router disconnects the source of undesired traffic by reinstalling the filter rule for the rest of T_{long}.

<u>Policy Module</u>

Though all end-hosts will have their own unique policy modules, the victim’s policy module is of particular interest to the implementation of the AITF protocol because good traffic is not always mutually exclusive with malicious traffic. The victim needs to decide whether or not sacrificing some good traffic for the sake of filtering out malicious traffic is viable. Implementing a policy module will be discussed in more detail in *Section 1.3.5*.

1.3.3 Actors

<u>Victim (*V*)</u>

The end-host that is the target of the DoS attacks is the victim. The victim will be referred to as *V* for the rest of the paper. It is assumed that *V* is an AITF-enabled end-host. *V*’s policy module will decide when to implement AITF, and when escalation is appropriate. *V* will be represented by a virtual machine (*VM1*), and will inherit *VM1*’s IP address.

<u>Attacker (*An*)</u>

The end-hosts that are the perpetrators of the DoS attacks are the attackers. The attackers will be referred to as *A1* and *A2* for the rest of the paper. It is assumed that *A* is an AITF-enabled end-host. However, it may either be a cooperative, or an uncooperative end-host. *A* will execute a UDP flooding attack of * packets/second directed at *V*. *A1* and *A2* will be represented by *VM2* and *VM4*, and will inherit their IP addresses.

Legitimate Host (*L*)

The legitimate host is an end-host that sends *V* good traffic that adheres to the criteria set forth by *V*'s policy module. The legitimate host will be referred to as *L* for the rest of the paper. *L* will send UDP datagrams to *V* at a rate of * packets/second. *L* will be represented by *VM5*, and will inherit *VM5*'s IP address.

Inter-route Routers (*R*)

The inter-route router is a router that comprises part of the flow between A_{gw} and V_{gw}. The inter-route router will be referred to as *R* for the rest of the paper. *R* will be represented by *VM3*, and will inherit *VM3*'s IP address.

Victim's Gateway Router (V_{gw})

V's border gateway router will be referred to as V_{gw} for the rest of the paper. It is assumed that V_{gw} is an AITF-enabled router. V_{gw} will accept filter requests from victim hosts, and send a corresponding filter request to the attacker's gateway router. V_{gw} will keep track of its current implemented filter rules by recording them in a local *filter table*. Should any incoming packets have a flow that is identified by a filter rule, then the packet will be dropped.

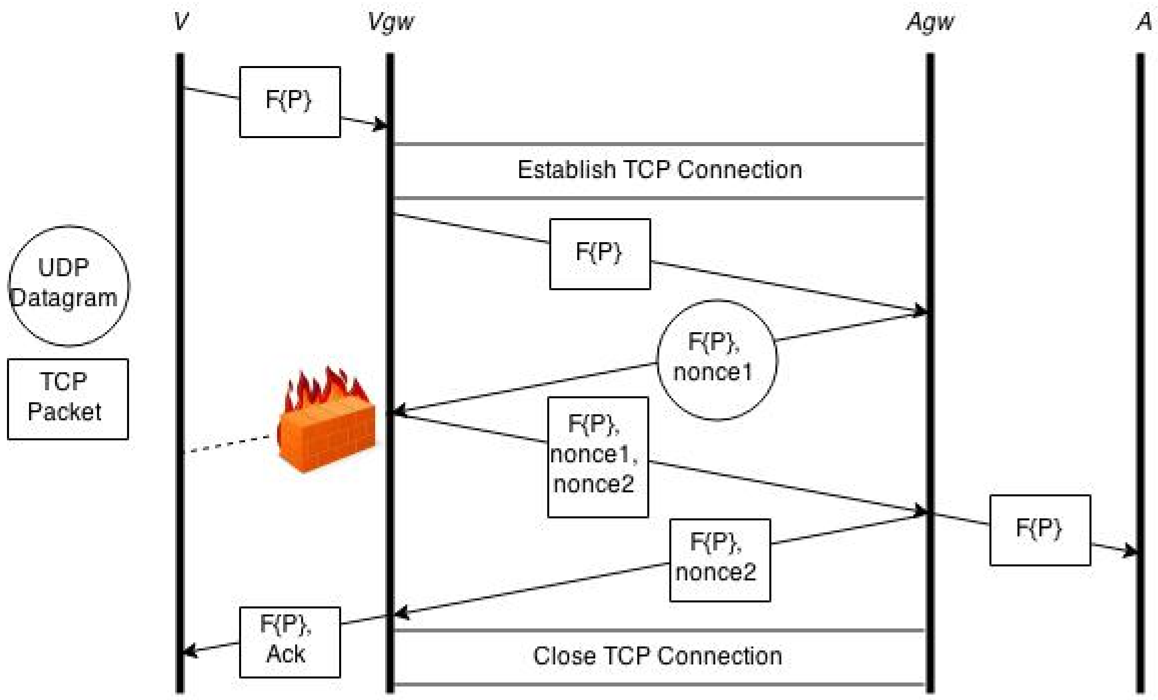
Filters that have been recorded in the filter table can last up to T_{tmp}. Once the filters are removed from the filter table, they are recorded in V_{gw}'s shadow filter table for up to T_{long}. V_{gw} will be represented by a laptop, and so will inherit its IP address.

Attacker's Gateway Router (A_{gw})

A's border gateway router will be referred to as A_{gw} for the rest of the paper. A_{gw} may be an AITF-enabled router or a legacy/ uncooperative router. A_{gw} can accept filter requests from V_{gw} and set up filter rules. A_{gw} will keep track of its current implemented filter rules through its local filter table. Filter rules that have been recorded in the filter table can last up to T_{tmp} seconds. Once the filter rules are removed from the filter table, they are recorded in A_{gw}'s *shadow filter table* for up to T_{long} minutes. A_{gw} will be represented by a laptop, and so will inherit its IP address.

1.3.4 The AITF Protocol

Figure 1: The AITF Protocol's 4-Way Handshake



1. *V*'s policy module determines an undesired flow (*F*). *V* sends a filter request to V_{gw}. The filter request is a TCP message with *F* as its contents.
2. V_{gw} immediately installs a local temporary filter rule that blocks all incoming traffic of *F*. This filter rule lasts up to T_{tmp} seconds. Once the temporary filter rule is removed, it is placed in the local *shadow filter table* for T_{long} minutes.
3. V_{gw} establishes a TCP connection with A_{gw}. V_{gw} sends a filter request to A_{gw}. The filter request is a TCP packet with *F* as its contents.
4. A_{gw} sends UDP datagrams containing *F* and *nonce1* to *V*.
5. V_{gw} intercepts the UDP datagram from A_{gw} and sends A_{gw} a TCP message containing *F*, *nonce1*, and *nonce2*.
6. A_{gw} first verifies that *nonce1* is correct. If so, it proceeds to verify that the R value corresponding to its IP address in the RR shim of the TCP packet from the the previous step is correct. If the R value is incorrect, then A_{gw} sends a TCP packet to V_{gw} with the correct R value corresponding to its IP address in the RR shim, and closes its TCP connection with V_{gw}. The AITF protocol ends here if the R value is incorrect. Otherwise, A_{gw} immediately installs a local temporary filter rule that blocks flow *F*. This filter rule lasts up to T_{tmp} seconds. Once the temporary filter rule is removed, it is placed in the local shadow filter table for T_{long} minutes.
7. A_{gw} then sends a TCP packet containing *F* and *nonce2* to V_{gw}. V_{gw} verifies that *nonce2* is correct, and removes the filter rule for *F*, if it is still up. V_{gw} will also send a TCP packet to *V* acknowledging that it has performed AITF and blocked the undesired flow. V_{gw} then closes its TCP connection with A_{gw}. A_{gw} sends *A* a filter request to stop for T_{long} minutes. This filter request is a UDP datagram with *F* as its contents. If A_{gw} detects *A* sending traffic of *F* while *F* is still recorded in the local *shadow filter table*, A_{gw} begins dropping *all* of *A*'s traffic.

8. If V_{gw} fails to negotiate the installation of a filter rule for F with A_{gw} , then it perform escalation, and repeat *Steps 2* to *8* with the next available router in *F*.

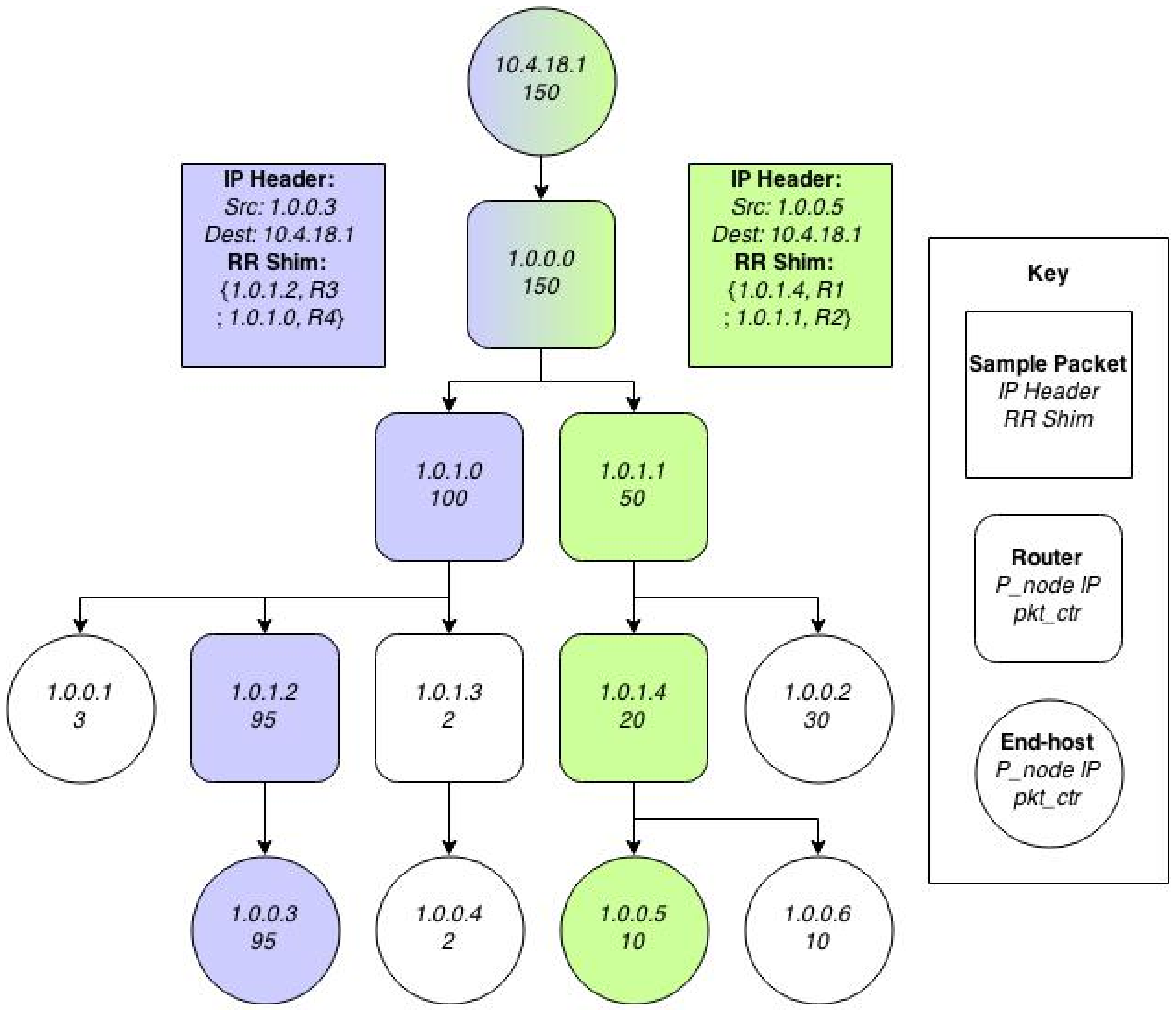
The protocol outlined above is indicative of how the AITF protocol will proceed, should *A* and A_{gw} be cooperative and not perform any form of spoofing. Variants of the AITF protocol will be addressed in *Section 2.1.4*, where different configurations of the actors will allow escalation and the AITF protocol's resistance to *RR spoofing* and *Lying* to be tested.

1.3.5 Implementing a Policy Module

The simplest way to implement a policy module is to define a new data type, which we'll call *PM_node*. *PM_node* records information indicating its *IP address* (with its *R value* if the node represents a router), a *packet counter*, which counts the number of packets with this IP address in its flow, and *pointers* to its children nodes. This means that the policy module will store all of its data in a *tree structure*. A packet counter of each node is supposed to equal the sum of its children's packet counters. *V*'s netfilter will increment the counter of each node that is discovered to be in the flow of an incoming packet with an RR shim (refer to *Figure 2*). The policy module will use the tree structure to find an undesired flow if there is one, while also periodically (every 25 milliseconds, for instance) resetting the structure.

Should the packet counter of the PM_node representing *V* exceed a predetermined limit, then *V* will consider itself to be the target of a DDoS attack. *V* will then iterate through its PM_node tree, checking their packet counters to see if the amount of traffic recorded by the node is acceptable.The process of finding an undesired flow is similar to walking a path from the root to a leaf in a tree, and the path will represents the undesired flow. If an undesirable quantity is found, and none of the upstream actors have undesirable quantities, then escalation is performed.

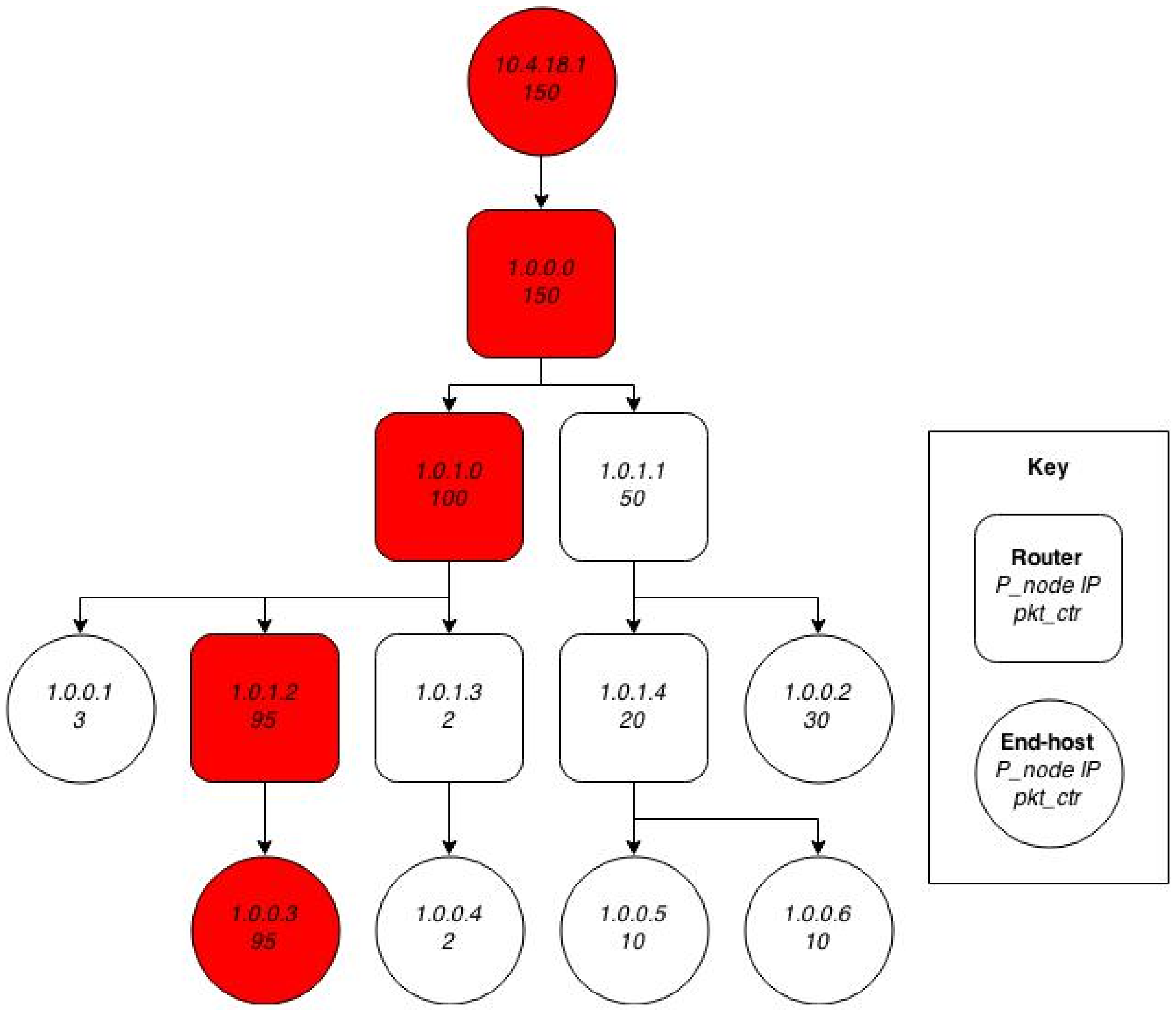
Figure 2: Policy Module's PM_node Tree



The packet counter allows *V* to identify undesirable flows, from which a malicious host or a subnet of hosts may be attacking it. Shown in *Figure 2* are two examples of incoming packets. When the purple packet arrives at netfilter, the counters of the purple color nodes will be incremented by 1; when the green packet arrives at netfilter, the counters of the green nodes will be incremented by 1.

Within the current time frame, a total of 150 packets have been inspected by *V*'s netfilter. 100 packets were found to have been forwarded by *1.0.1.0*, 50 packets were found to have been forwarded by *1.0.1.1*, and the total of these two equal their parent's packet counter, 150.

Figure 3: Identifying an Undesired Flow



Let us say that *V* has a rule in its policy module that identifies the unwanted flow { *1.0.0.3* : *1.0.1.2* : *1.0.1.0* : *1.0.0.0* : *10.4.18.1* }. The policy module will start from the root, and walk to the bottom. If the current node has a packet counter above the limit (its value defined by policy module), it will choose its children node with largest packet counter as the next step. The path will represent an undesired flow. If it walks to a node with a lot of children node, the node will be identify as IP spoofing, and therefore the path from the root to this node will be returned as undesired flow.

2 Evaluation Criteria

2.1 Performance

The AITF implementation's success will be based on three measurable variables.

2.1.1 Round Trip Time

The *ICMP ping probe* was used to measure the additional latency introduced to system due to the incorporation of the AITF protocol. The *mean round trip time* (*RTT*) of a packet is defined to be the amount of time taken for a signal to be sent plus the length of time it takes for an acknowledgement of that signal to be received. The difference of RTT values between packets without the RR shim and packets with the RR shim indicates the total amount of time taken to insert, modify, and insert the RR shim in AITF protocol.

2.1.2 Filtering Response Time

The *filtering response time* (T_{fr}) is defined to be the time that elapses from the moment *V* sends a filtering request against an undesired flow, until *V* stops receiving the flow. T_{fr} will be largely affected by the delay incurred by the additional overhead resulting from implementing the AITF protocol.

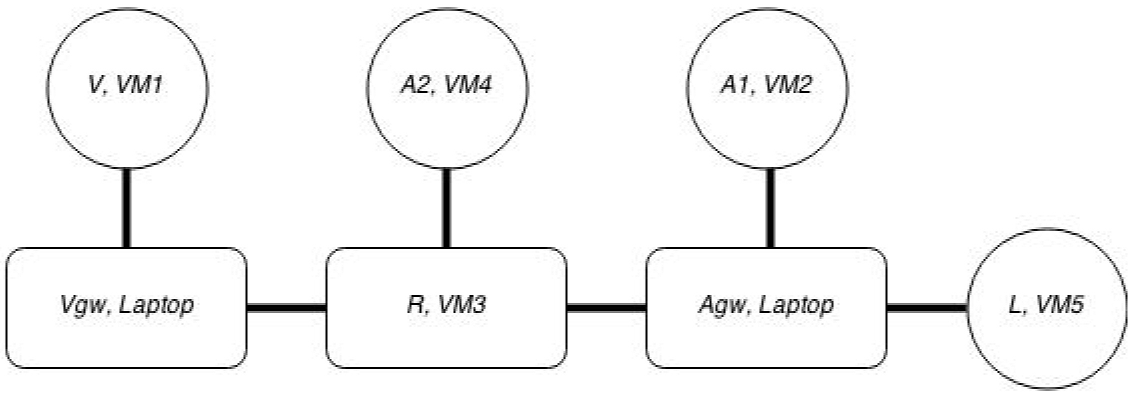
2.1.3 Relative Difference

The *relative difference* is the relative difference between the rate of packets being sent by *Ln* (*r1*) and the rate of packets being received by *V* (*r2*). This may be calculated through the following equation: $relative\ difference = |\frac{r2-r1}{r1}| * 100\%$. Should the relative difference be less than 5% after T_{fr} has passed, then the AITF protocol will be considered to have been successful. It should be noted that *V*'s bandwidth will be measured not in Mbps, as is traditional, but in packets/ second. Measuring *V*'s goodput in packets/ second is a simple abstraction to make, and will greatly simplify the process of gathering and analyzing data.

2.1.4 Criteria

The criteria for success will be obtained through tests based on the setup in *Figure 4*.

Figure 4: General Actor Configuration



<u>Normal Traffic</u>

Scenario 0: Normal traffic from L to V without AITF.

The AITF protocol will not have been implemented. **This scenario will serve as a baseline to compare the results from other scenarios.**

Scenario 1: Normal traffic from L to V with AITF.

No filtering will be carried out, but the AITF protocol will have been implemented. **This scenario will serve to indicate the effect of implementing the AITF protocol on throughput may be observed by comparing the results obtained to the results obtained in *Scenario 0*.**

<u>Malicious Traffic</u>

Scenario 2: Malicious traffic from A1 to V with AITF. Normal traffic from L to V. Cooperative AITF-enabled A1 adheres to the AITF protocol.

The standard AITF protocol will be carried out, as is detailed in *Section 1.3.4*. *A1* resumes its attacks after T_{long}. A successful implementation of this scenario will result in *L*'s traffic to *V* having a relative difference of ~0%, and *A1*'s traffic to *V* having a relative difference of 100%. **This scenario will serve to indicate the standard functionality of the AITF protocol with an attacker that does not wish to be disconnected.**

Scenario 3: Malicious traffic from A1 to V. Normal traffic from L to V. Uncooperative AITF-enabled A1 ignores filter requests (Possibly Legacy). Cooperative AITF-enabled A_{gw}.

The standard AITF protocol will be carried out, as is detailed in *Section 1.3.4*. *A1* will be disconnected by A_{gw} in *Step 7* of *Section 1.3.4*. A successful implementation of this scenario will result in *L*'s traffic to *V* having a relative difference of ~0%, and *A1*'s traffic to *V* having a relative difference of 100%. **This scenario will serve to indicate the standard functionality of the AITF protocol with either a legacy attacker or an attacker that doesn't mind being disconnected.**

This scenario also covers how IP spoofing would be handled, should it have been implemented in *V*'s policy module.

<u>Malicious Traffic - Escalation</u>

Scenario 4: Malicious traffic from A1 to V. Normal traffic from L to V. Uncooperative AITF-enabled A_{gw} ignores filter requests (Possibly Legacy).

A variant of *Step 4* of *Section 1.3.4* will be performed. After the filter request times out, V_{gw} will perform escalation, and attempt to perform AITF with *R* to filter out undesired traffic from A_{gw}. A successful implementation of this scenario will result in *A1*'s traffic to *V* having a relative difference of 100%. *L*'s traffic to *V* should also have a relative difference of 100%, as escalation will have been performed in a successful test. **This scenario will serve to indicate the results of performing escalation in the AITF protocol against a legacy router.**

Scenario 5: Malicious traffic from A1 to V. Normal traffic from L to V. Uncooperative AITF-enabled A_{gw} lies about installing filter.

A variant of *Step 4* of *Section 1.3.4* will be performed. After the filter request times out, V_{gw} will perform escalation, and attempt to perform AITF with *R* to filter out undesired traffic from A_{gw}. A successful implementation of this scenario will result in *A1*'s traffic to *V* having a relative difference of 100%. *L*'s traffic to *V* should also have a relative difference of 100%, as escalation will have been performed in a successful test. **This scenario will serve to indicate the results of performing escalation in the AITF protocol against a compromised router.**

Scenario 6: Malicious RR spoofed filter requests from uncooperative AITF-enabled A2, posing as A_{gw}, to V_{gw}. Normal traffic from L to V. Cooperative AITF-enabled A_{gw}.

A variant of *Step 6* will be performed. V_{gw} will perform escalation, and attempt to perform AITF with *R* to filter out the undesired filter requests from *A2*. **This scenario will serve to indicate the AITF protocol's resistivity to RR spoofed attacks.**

2.2 Security

The security components addressed by the AITF protocol are authenticity and availability.

2.2.1 Authenticity

The RR shim will be used to improve the authenticity of the AITF protocol. Also, during the 4-way handshake, by specifically having V_{gw} intercept the UDP datagram sent to *V* to obtain the correct nonce, the authenticity of the AITF protocol was improved.

2.2.2 Availability

Maintaining the availability of good traffic to *V* and denying the availability of *V* to *A* are the main premises for implementing the AITF protocol.

3 Results

* All times are in milliseconds unless stated otherwise.

Scenario 0:

Two trials of ICMP ping probes were sent out from *L* to *V*. The trials lasted approximately 50 seconds each. AITF had not yet been implemented.

Results

Table 0: Normal ICMP Probes

Time (ms)	Mean (ms)	Min. (ms)	Max. (ms)	MDev.	Packets	Interval (sec)	Pkt. Loss
55246	1.659	0.751	113.431	1.581	32284	0.001	0%

Relative Difference

The relative difference is indicated by the % of packets lost. In both trials, the relative difference was 0%. This indicates that the connection between the actors used is stable.

Scenario 1:

An ICMP ping probe was sent out from *L* to *V*. The trial lasted approximately 50 seconds. AITF had been implemented, but the policy module had not yet been activated.

Results

Table 1: ICMP Probes with RR Shim

Time (ms)	Mean (ms)	Min. (ms)	Max. (ms)	MDev.	Packets	Interval (sec)	Pkt. Loss
53682	2.688	1.155	16.86	1.207	19058	0.001	0%

Relative Difference

The recorded relative difference was 0%. This indicates that the connection between the actors and the process of inserting and removing the RR shim is stable, and doesn't cause packets to be dropped.

RTT

The difference in mean RTT values between *Scenario 0* and *Scenario 1* indicates that implementing the AITF protocol does incur a performance hit. The difference in mean RTT values is 2.688 - 1.6815 = 1.0065.

Implementing the AITF protocol increased the RTT of packets by approximately 1 millisecond. According to the literature, internet RTT values in reality usually range from 50 to 200 milliseconds, so even though a hit to performance is taken, it is an negligible increase in latency, especially in light of the benefits the AITF protocol brings.

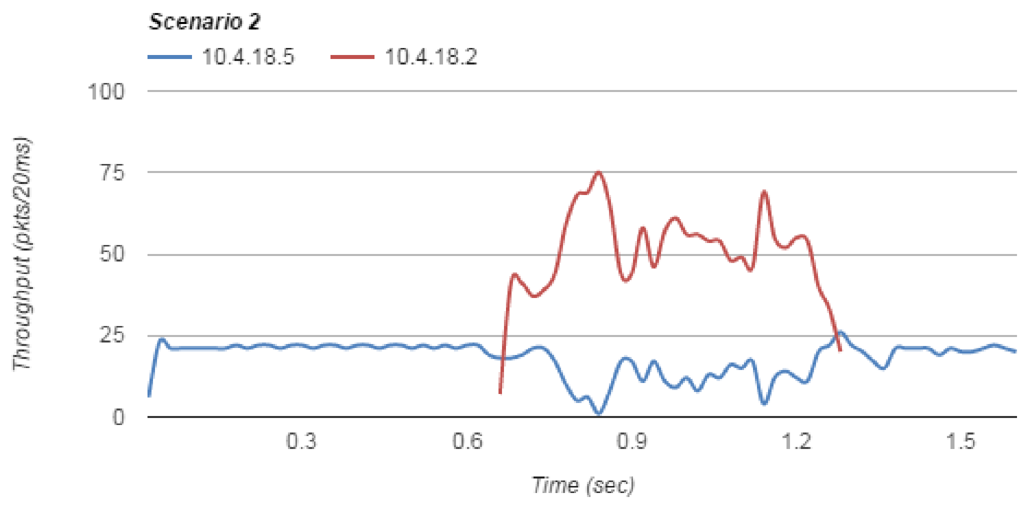
Scenario 2:

An ICMP probe was used in *Scenario 2* to test a compliant AITF-enabled end-host's response time. An ICMP probe was sent out in intervals of 20 ms over the span of 2 seconds, from *L*(*10.4.18.5*) to *V*. Another ICMP probe was sent out in intervals of 20 ms around 0.6 seconds after the first ICMP probe was sent out, from *A1*(*10.4.18.2*) to *V*. *V*'s police module defined any incoming flow of packets of over 50 pkts/20 ms to be undesirable.

The displayed results were obtained from a setup of *Scenario 2*.

Results

Plot 0: Scenario 2; A1 Traffic versus AITF



Relative Difference

Prior to receiving *A1*'s traffic, *L*'s traffic was arriving at *V* at a steady rate of 22 pkts/ 20 ms. Once *A1* started performing a UDP flooding attack on *V*, *L*'s traffic became unstable and would dip whenever *A1*'s traffic spiked. However, as may be seen in *Plot 0*, the traffic prior to, and after *A1*'s attack is the same, at 22 pkts/ 20 ms.

Filtering Response Time

The filtering request will be considered to have been sent by *V* at the 0.75 second mark, and successfully fulfilled at the 1.3 second mark. The filtering response time done through these calculations is thus 550 ms.The filter response time in this scenario is much greater than the actual filtering response time because of using the tool ping as the packet sender. It should be noted that at the time this scenario was performed, the server the VM's were running on was being heavily used, so performance was not as fast as it could have been.

Conclusion

The test that was run to simulate *Scenario 2* was considered to have been a **success** as the relative difference is not significant, and all malicious packets sent by *A* were filtered out after 550 ms.

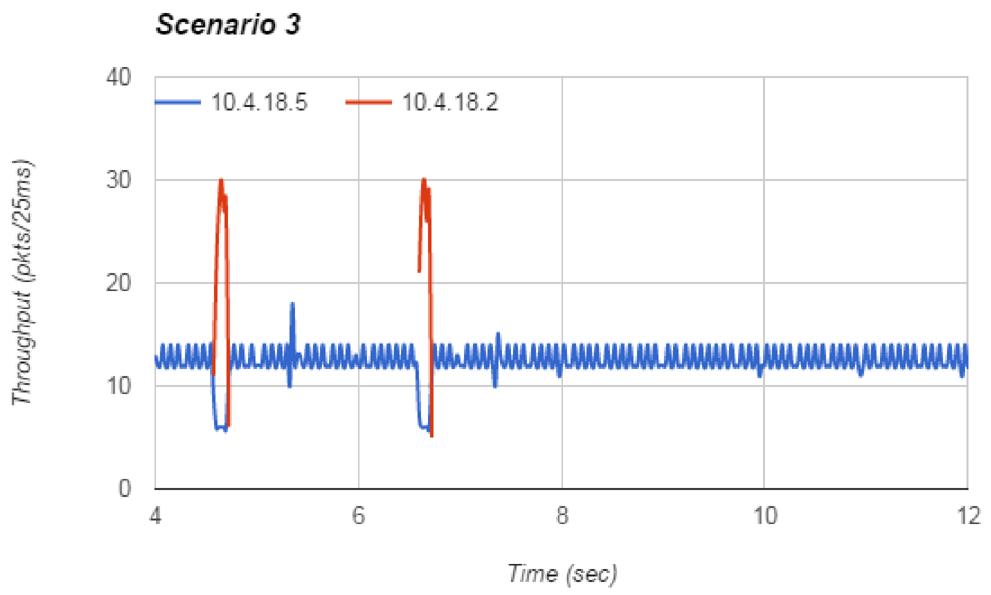
Scenario 3:

In *Scenario 3*, UDP datagrams were sent at a rate of 13 pkts/25 ms from *A1* to *V*. UDP datagrams were also sent at a rate of 30 pkts/25 ms from *L* to *V*. *V*'s policy module defined any incoming flow of packets of over 25 pkts/25 ms to be undesirable. *A1* began sending packets at 100 pkts/25 ms at the 4.5 second mark. T_{tmp} was set to be 2 seconds in this scenario, to emphasize the effectiveness of AITF against an uncooperative attacker.

The displayed results were obtained from a setup of *Scenario 3*.

Results

Plot 1: Scenario 3; A1 Traffic versus AITF (Non-Compliant)



Relative Difference

As is indicated in *Plot 1*, the relative difference for *L*'s traffic arriving at *V* before and after AITF was used to filter out *A1*'s traffic was 0.

Filtering Response Time

The filtering request was sent by *V* at the 4.6 second mark, and completed at the 4.75 mark. The large improvement in performance was due to tests being carried out with a udp packet sending program, instead of ping. The filtering response time done through these calculations is thus 125 ms, which is close to the actual filtering response time. The second burst in malicious traffic in *Plot 1* indicates when A_{gw} removes its temporary filter rule, after T_{tmp}. Thereafter, *A1*'s traffic was dropped by A_{gw} for T_{long} , due to its recorded path to *V* existing in *V*'s shadow filter table. The interval between the spikes indicates T_{tmp}

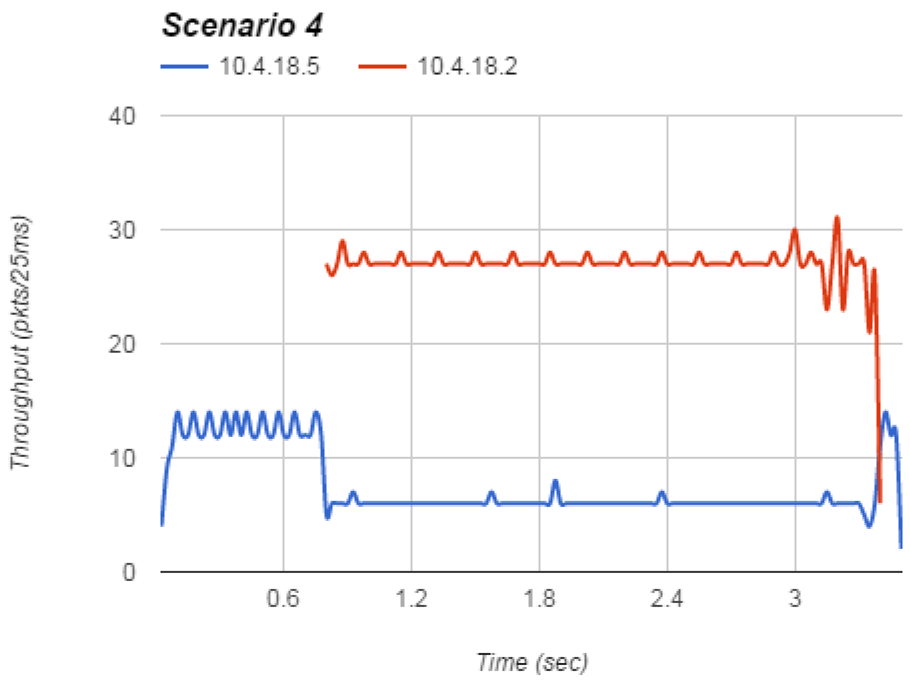
Conclusion

The test that was run to simulate *Scenario 3* was considered to have been a **success** as the relative difference is not significant, and all malicious packets sent by *A1* were filtered out for at least T_{long} after the second burst of malicious traffic from *A1* was received by *V*.

Scenario 4:

Results

Plot 2: Scenario 4; Traffic versus Complete Escalation



Relative Difference

As may be seen in *Plot 2*, *V* ceases to receive traffic from both *A1* and *L* after a total test run of 3.6 seconds.

Filtering Response Time

The filtering request was sent at the 0.8 second mark, and the AITF protocol successfully executed escalation to block out the undesirable flow (including *L*'s non-malicious traffic) at the 3.6 second mark. The filtering response time was thus 2.8 seconds. The TCP's timeout mechanics were the reason behind the increased filtering response time, as TCP was used as part of the AITF protocol's 4-way handshake. The undesired rate of incoming packets was set to 25 pkts/25 ms.

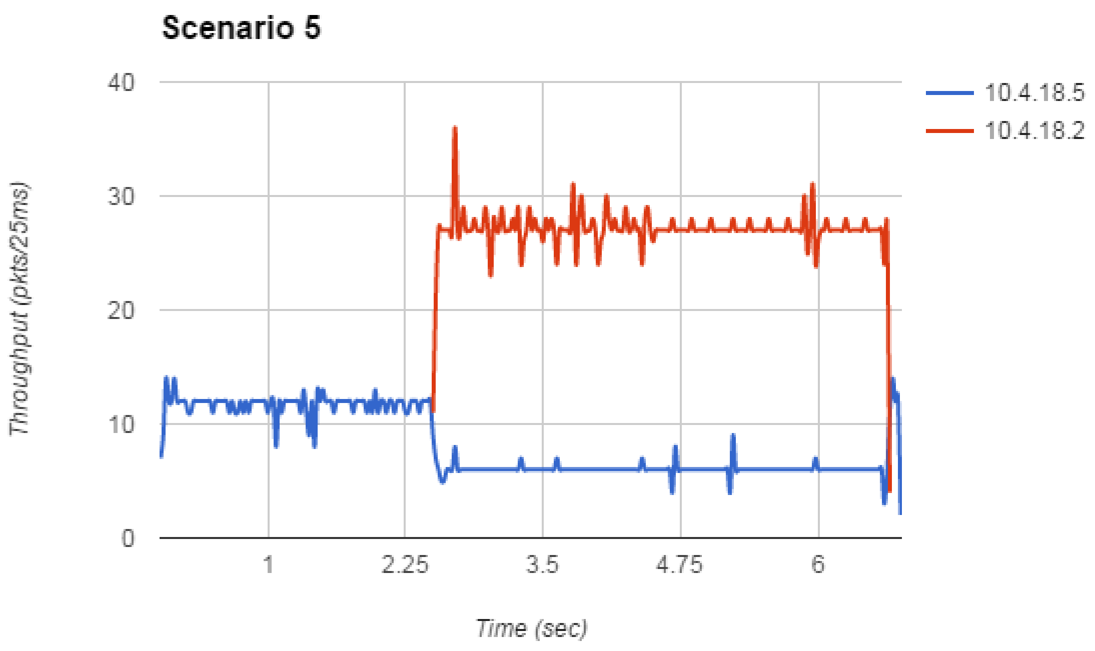
Conclusion

The test that was run to simulate *Scenario 4* was considered to have been a **success** as the relative difference of both sources of traffic were 0% at the end of the test. It should be noted that the amount of time taken to filter traffic via escalation was significantly higher than the amount of time taken to filter traffic through the standard AITF protocol. This was because V_{gw} was trying to establish a connection with A_{gw} to follow through with AITF via TCP. However, V_{gw} was able to escalate to *R*, and both innocent and malicious flows were filtered out.

Scenario 5:

Results

Plot 3: Scenario 5; The Lying Router versus Complete Escalation



Relative Difference

As may be seen in *Plot 3*, *V* ceases to receive traffic from both *A1* and *L* after a total test run of 6.8 seconds.

Filtering Response Time

The filtering request was sent at the 2.5 second mark, and the AITF protocol successfully executed escalation to block out the undesirable flow (including *L*'s non-malicious traffic) at the 6.8 mark. The filtering response time was 4.3 seconds. The undesired rate of incoming packets was set to 25 pkts/25 ms.

Conclusion

The test that was run to simulate *Scenario 5* was considered to have been a **success** as the relative difference of both sources of traffic were 0% at the end of the test. It should be noted that the amount of time taken to filter traffic via escalation in this scenario was significantly higher than the amount of time taken to filter traffic in all other scenarios due to two main reasons. The first reason was mentioned in the analysis of the previous scenario. TCP, when involved in situations like these, adds significant delay due to its timeout mechanics. The second reason is that A_{gw} lying about having installed a filter sabotages the standard AITF protocol by dissuading V_{gw} from escalating, while A_{gw} secretly lets *A1*'s traffic through. Due to this, escalation was forcefully executed by *V*'s policy module once it had detected that A_{gw} had lied twice about filtering *A1*'s traffic to *V*.

Scenario 6:

Simple tests against RR spoofing were performed. Instead of a secure hash function, a simple XOR cipher was used. This was done to ensure that the R value in the RR shim performed its function properly. Were a better level of security to be desired, then another hash function, like DES, 3DES, MD5, or any of the publically published ones could be implemented to protect AITF against RR spoofing brute force attacks.

4 Conclusion

Our implementation was able to successfully block 100% of undesired flows in the simulated environment. Non-malicious traffic will not be impacted adversely by the AITF protocol, if escalation is not performed. If escalation is executed according to an effective policy module, then though some non-malicious traffic will be filtered out, the lack of wasted resources will benefit the host more than the host will suffer to lose.

A significant bottleneck to the implemented AITF protocol design was the use of TCP. Though TCP does provide the advantage of guaranteeing the delivery of filter requests, the delay incurred by TCP's timeout mechanics and SYN packets may prove to be infeasible in a real-world application. Other facets of this project which may be further improved include scale testing, to guarantee that a real-world internet implementation will prove to remain effective in the long run; improved hash function security to prevent brute force attacks from compromising the effectivity of the AITF protocol; and security against IP spoofing.

<u>Improvements</u>

<u>Design Document</u>

+ Paper organization and structure underwent massive rehaul
+ Added an inter-route router to the setup
+ Added assumptions section
+ Added diagrams
+ Organized threat model into distinct subsections
+ Added section on netfilter to elaborate on adding shim
- Removed multiple attackers and implemented bandwidth throttling
- Removed technical function definitions and elaborated on implementation
- Removed phase of AITF where hosts tells its gateway router if it is AITF-enabled

<u>Evaluation Criteria</u>

+ Added more potential scenarios
+ Added RTT and filtering response time as criteria for success
+ Clarified criteria for success
+ AITF protocol was divided into atomic steps
+ Added section on implementing *V*'s policy module
- Removed sections on T_{tmp} and T_{long}

<u>Results</u>

+ Conclusions and discussion were combined with data results
+ The results section was split according to scenarios rather than information categories
- Removed traffic spoofing scenario
- Removed tables and replaced with visual graphs