Stockade Performance and Stability Testing

Adam Black, Malcolm Robb
Centre for Advanced Internet Architectures. Technical Report 070402B
Swinburne University of Technology
Melbourne, Australia
adamblack@swin.edu.au
mrobb@swin.edu.au

Abstract- Stockade is a network-layer spam-mitigation software designed to prevent spammers from consuming excess bandwidth on a mail server. Several experiments have been conducted on Stockade to assess the performance and stability of the software. The first experiment revealed that Stockade could achieve approximately a 93% reduction in spam traffic under our test conditions. Stockade's automatic rehabilitation feature was shown to operate accordingly and allowed blacklisted hosts to eventually become unblacklisted over a period of time. Further testing into the rehabilitation feature showed that a client re-blacklisted during their rehabilitation was eventually unblacklisted by Stockade. Lastly we examined Stockade's method of blocking an entire subnet which operated effectively. We also verified that neighbouring blacklisted/greylisted IP entries would be merged automatically into a single blacklist entry. In section VII I have mentioned possible future endeavors for testing Stockade.

I. Introduction

This document has been prepared to inform the reader about several experiments conducted using Stockade version 0.2 [1]. The experiments were designed to test the stability and performance of the Stockade mail filter.

Stockade is a network-layer spam-mitigation package designed to work in conjunction with mail filtering programs, such as SpamAssassin [6] and SpamBayes [7]. Stockade filters packets at the TCP layer and can prevent spam e-mails from consuming excess bandwidth on a mail server. Stockade uses probabilistic rejection [9] to determine whether an incoming SMTP connection should be accepted or rejected.

A feature of Stockade is the ability to automatically rehabilitate clients who have previously been registered as spammers. This rehabilitation feature means that an IP address registered as a spam source with a blocking probability of 1.0 will eventually be allowed to send e-mails without any chance of being blocked, given they do not send spam during the rehabilitation period. As a result a host who is constantly flooding a mail server with spam will have few of the connection attempts accepted, depending on the spammers sending rate. The rehabilitation period is dependent on the MetricDecayFactor, which determines how much time must elapse prior to the drop probability of a client halving.

The main contents of this document have been outlined in the following paragraphs.

Section II is a brief introduction into Stockade and explains the configurable properties which have been varied throughout the testing.

In Section III I have developed a contrived yet straightforward method of testing and gauging the performance of Stockade on a mail server. The experiment is designed to give a quantitative measure to the amount of bandwidth consumed with and without Stockade in use.

Section IV examines the rehabilitation feature of Stockade. The experiments are designed to test whether Stockade's rehabilitation process works as alleged.

Section V revisits the rehabilitation aspect of Stockade, and observes the effect of re-registering a client as a spammer during the rehabilitation period.

In Section VI I have explored a method of blacklisting/greylisting an entire subnet (or range of hosts) known as Poisoning the Neighbourhood. In addition I have tested a feature of Stockade known as Neighbourhood Aggregation. Briefly explained, Neighbourhood Aggregation combines adjacent IP address entries, which have been blacklisted or greylisted, and merges them into a single entry. The benefit of merging adjacent entries is that the server will have less processing overhead, and fewer blacklisted and/or greylisted entries

Section VII covers the limitations of the testing methods used.

Section VIII concludes the report and summarises the results.

References are listed at the end of this document.

II. Background on Stockade

This section will provide a basic overview into the usage of Stockade. I will begin in subsection A by demonstrating how an IP addresses can be registered as spammer. In subsections B and C I will commentate the various properties presented in Stockade's configuration file which are mentioned within this document.

Throughout this report there are references to blacklisted and greylisted IP addresses. Blacklisted IP addresses are ones which are not allowed SMTP communication with the mail server and have a spam rating or blocking probability of 1.0. Greylisted hosts are on occasions allowed SMTP communication with the server and will have a blocking probability less than

1.0. Also note I will be referring to the terms: 'dropping probability' and 'spam rating' synonymously.

A. Basic Stockade Usage

Stockade provides an executable file and a Perl module which allows the user to blacklist or greylist IP addresses. The usage is outlined below:

Using command line executable:

register_spam(Tag, IP[/Network bits], Metric)

Using Perl module:

registerSpam(Tag, IP[/Network bits], Metric)

The Tag argument is a spam classification tag and may be used to normalise metrics by the blacklist daemon. The IP tag is the range of IP addresses to blacklist or greylist. If the [/Network bits] keyword is omitted a single IP entry is blocked, if the network bits are included an entire subnet is blocked. The Metric is the initially blocking probability to be applied to the IP address or range.

B. Rehabilitation and the Configurable Parameters

A feature of Stockade, known as automatic rehabilitation; is the ability for blacklisted and greylisted IP addresses to eventually become completely unblocked for SMTP communication with the mail server. If an IP address is blacklisted by Stockade they will not be allowed to establish an SMTP session with the mail server until their dropping probability falls to a value below 1.0. Once the client's spam rating drops below 1.0 they will be able to communicate with the SMTP server, however the probability of a successful connection is determined by: $1 - \text{Spam rating}$. When the spam rating is 0, the client will not be blocked by Stockade under any circumstances.

The rehabilitation parameters I will discuss include the following:

- RehabilitationHeartbeatInterval
- RehabilitationIterationsPerHeartbeat
- UpdateTickTime
- MetricDecayFactor
- MinDropProbability

Rehabilitation is implemented by reducing each senders drop probability according to an exponentially decaying function with the half life determined by the MetricDecayFactor property. Simplified, this means that after the MetricDecayFactor time has elapsed the probability of dropping new SMTP session halves. When the spam rating is less than the MinDropProbability the IP entry is completely unblacklisted by Stockade.

Rehabilitation calculations are performed sequentially on each IP address entry. The minimum interval between the calculations is determined by the RehabilitationHeatbeatInterval. At every heatbeat multiple IP address entries can be rehabilitated, this is determined by the RehabilitationIterationsPerHeartbeat property.

The UpdateTickTime is the minimum number of seconds between rehabilitation calculations

C. Dropping Packets

There are two packet filtering strategies which can be used by Stockade, Ipfw [4] rule based filtering and divert socket based filtering. The packet filtering strategy is configurable via the PacketFilterType parameter. Throughout the experiments Ipfw was used.

Ipfw has three methods of denying a client communication with a mail server. Stockade lets you configure which method will be used via the IpfwRejectAction property. The available options and their descriptions are shown below:

- Deny - Drop packets silently
- Reject - Send an ICMP unreachable response
- Reset - Send a RST packet.

The Reset method was used throughout the experiments. The diagram below illustrates a successful and unsuccessful connection attempt to the SMTP server at the TCP level.

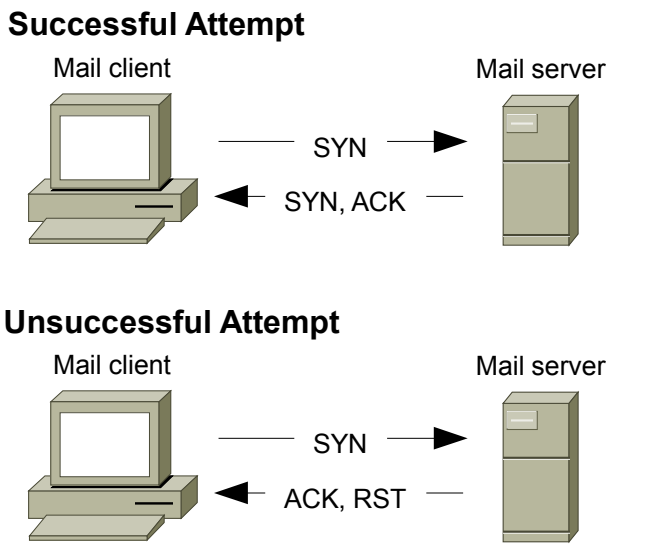

Figure 1: TCP Conversation Between Mail Client and Mail Server Using Reset IpfwRejectAction

III. Stockade Performance Analysis

A. Introduction:

The intent of this section is to demonstrate to the reader how much bandwidth can potentially be saved by implementing Stockade on a mail server which only uses content filtering to categorise and administer spam.

A real mail server will have a mixture of ham and spam, or solicited and unsolicited e-mails. Gathering and analysing detailed statistics about spam and ham sending patterns on a real mail server is not a trivial task. To simplify the testing of Stockade's performance (in terms of bandwidth savings) several assumptions had to be made. These are discussed under subsection F.

B. Aim:

- To test how stockade behaves with multiple entries in the Ipfw table.
- To approximate the amount of bandwidth that can be saved by using Stockade in conjunction with a mail filtering program, such as Procmail [8], SpamAssassin, or SpamBayes.

C. Method:

1) Equipment and Setup

- 2 x PC's running FreeBSD 6.2 [2] networked over a FastEthernet connection. They will be referred to as the mail client and mail server.
- The client had the following additional programs used throughout the testing:
  - smtpclient-mod 1.0.0 - Modified application from the FreeBSD ports collection, able to send e-mails bound to an alias IP address.
  - sendmsg2 – A script written to send e-mails repeatedly to the server using smtpclient and bound to various aliased IP addresses.
- The server had the following additional programs used throughout the testing:
  - sendmail 8.13.8.
  - checkmsg.pl – Perl script to check source IP address of an e-mail, and register predefined IP addresses as spammers with Stockade.
  - ipfw (as with FreeBSD 6.2 release).
  - stockade 0.2.
  - procmail 3.22.
- The client was configured with 20 IP addresses
- The server was configured with a single IP address.

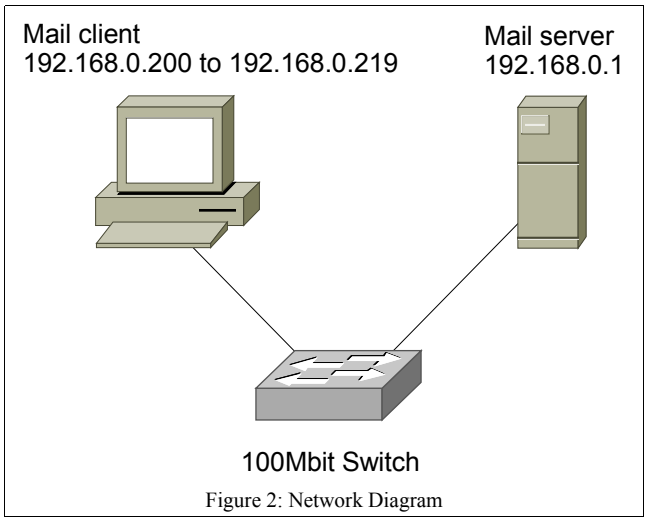

Figure 2: Network Diagram

2) Procedure – Experiment 1

- A MetricDecayFactor of 300 seconds was used.
- The client had 20 IP addresses configured.
- Identical e-mails of 1292 bytes in length were sent from the client to the server with an exponentially distributed random delay between sending attempts. $Delay \sim Exponential(\lambda), where\ \lambda = 20\text{msg}/sec$
- Each sending attempt by the client was bound to a different IP address in sequential order.
- A list of bad IP's (badips.txt) was created on the server with 4 senders predefined as spammers.
- E-mails received by the server were filtered by Procmail and a Perl script was executed on each e-mail
- The Perl script would register the sender as a spammer with a rating of 1.0, if the IP address matched an entry in the badips.txt file

Figure 3 displays a flowchart illustrating the testing process.

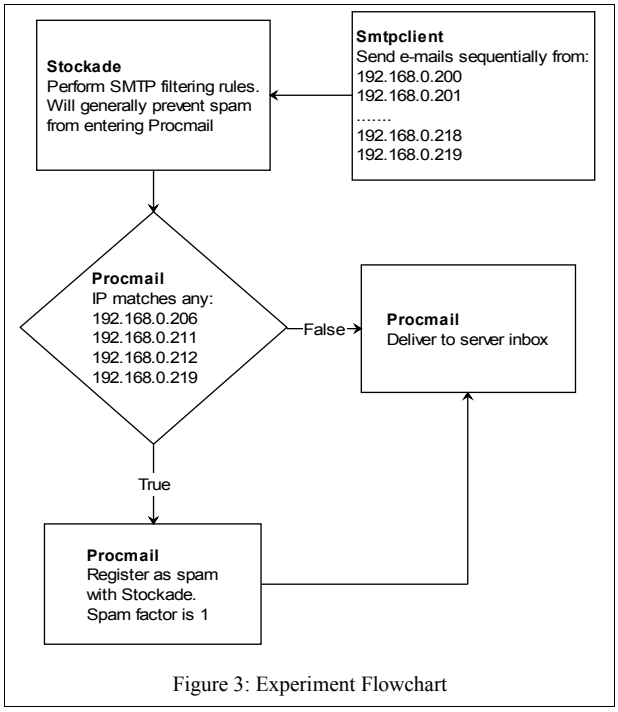

Figure 3: Experiment Flowchart

NB: In a real mail setup spam will likely be routed to a separate spam folder, or disregarded completely.

D. Results:

1) Analysis Method

- A tcpdump [3] was taken during a successful and unsuccessful attempt at contacting the mail server. The payloads were calculated in each case, and used to calculate bandwidth consumption.
- The log file created by sendmsg2 was analysed by a Python script to determine how many ham and spam messages were successfully sent.
- The results were plotted using Microsoft Excel.

2) Experiment 1 Results

The amount of incoming and outgoing TCP spam traffic is shown in Figure 4. The results illustrate the bandwidth usage with and without Stockade.

Below are the results from the analysed sendmsg2 log file. "Total [ham, spam] traffic" is the total amount of upstream and downstream traffic caused by ham/spam given Stockade was not in use. "[Ham, Spam] traffic using stockade" is the amount of TCP traffic generated due to ham/spam messages while stockade was in use.

Total ham traffic: 20112144 bytes

Total spam traffic: 5025527 bytes

Ham traffic using stockade: 20112144 bytes

Spam traffic using stockade: 339365 bytes

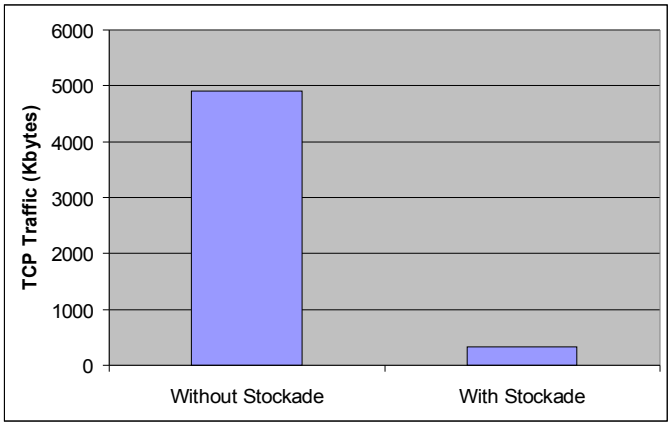

Figure 4: TCP Spam Traffic Consumption With Multiple Senders

E. Conclusion:

When Stockade was implemented on the mail server a substantial difference in traffic throughput can be seen by the graph shown in Figure 4. It is also notable that none of the ham traffic was blocked by Stockade.

F. Limitations:

Some of the limitations of the testing method used includes the following:

- The traffic profile on a mail server may vary significantly.
- Arrival behaviour of ham and spam was extremely simplified.
- Assumes 20% of e-mail is spam.
- All e-mails were a fixed length.
- All e-mails were sent sequentially, a better model may be concurrent sending attempts.
- Stockade was not tested with SpamAssassin.
- Many defining properties in stockade configuration file were not varied, such as: UpdateTickTime, RehabilitationHeartbeatInterval, RehabilitationIterationsPerHeartbeat, IpfwRejectAction, MetricDecayFactor etc.

IV. Rehabilitation Testing

A. Introduction:

A feature of Stockade is automatic rehabilitation, which reduces the blocking probability of blacklisted and greylisted hosts over time. The rate of the rehabilitation is determined by the MetricDecayFactor property located in the stockade.conf file.

B. Aim:

To test Stockade's rehabilitation process, and ensure that clients who have previously been sending spam will be rehabilitated by stockade over time.

C. Method:

1) Equipment and setup

- 2 x PC's running FreeBSD 6.2 networked over a FastEthernet connection. They will be referred to as the mail client and mail server.
- The client had the following additional programs used throughout the testing:
  - smtpclient 1.0.0.
  - sendmsg – A script written to send e-mails to the server at specified intervals using smtpclient.
- The server had the following additional programs used throughout the testing:
  - sendmail 8.13.8.
  - ipfw (as with FreeBSD 6.2 release).
  - stockade 0.2.

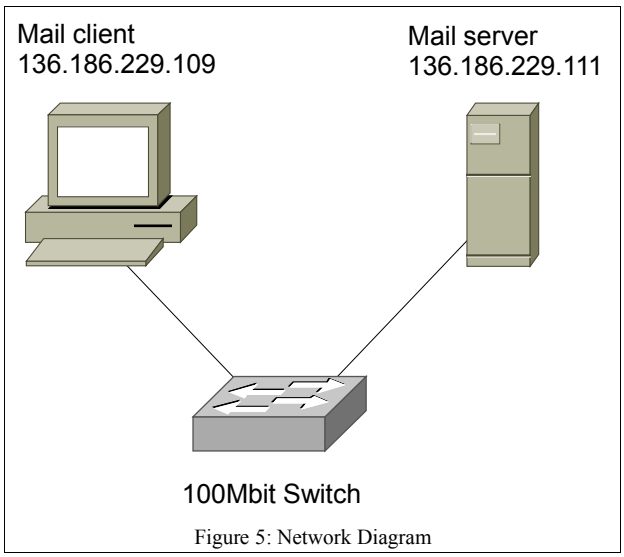

Figure 5: Network Diagram

2) Procedure

- The client's IP address was initially given a spam rating of 1.0 by Stockade so that all incoming e-mails from the client were initially rejected.
- Identical e-mail messages were sent (using sendmsg) from the client to the server at a certain rate, shown in Table 1

- The MetricDecayFactor property (in the stockade.conf file) was varied for Experiment 3.
- All successful and unsuccessful attempts were logged from the client using tcpdump.

Table 1 contains variables of interest for the three experiments conducted.

Message rate is how regularly messages were sent from the client to the server.

	E-mail rate (msg/sec)	MetricDecayFactor (mins)
Exp 1	2 msg/sec	5 mins
Exp 2	10 msg/sec	5 mins
Exp 3	42 msg/sec	2.5 mins

Table 1: Rehabilitation Test Variables

D. Results:

1) Analysis Method

- The tcpdump file was analysed and used to determine the number of successful and unsuccessful attempts according to the guidelines below:
  - A packet with an syn and ack flag set indicated a successful connection with the SMTP server.
  - A packet with an ack and rst flag set indicated an unsuccessful connection with the SMTP server.
- Two Python scripts were written to perform statistical analysis on the tcpdump files, one script performed a histogram average, while the other script performed a moving window average.
- The results were plotted using Microsoft Excel.

2) Experiment 1 Results

E-mail rate ~ 2 msg/sec.

MetricDecayFactor = 5 mins.

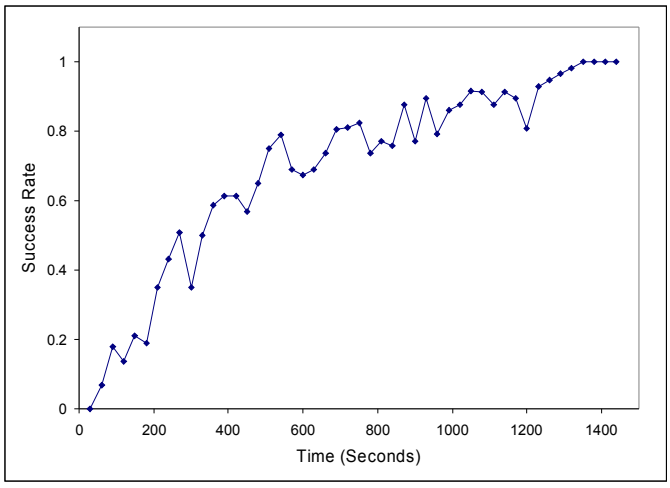

Figure 6:Rehabilitation Example, Histogram Average, Bucket Size = 30 secs

3) Experiment 2 Results

E-mail rate ~ 10 msg/sec

MetricDecayFactor = 5 mins

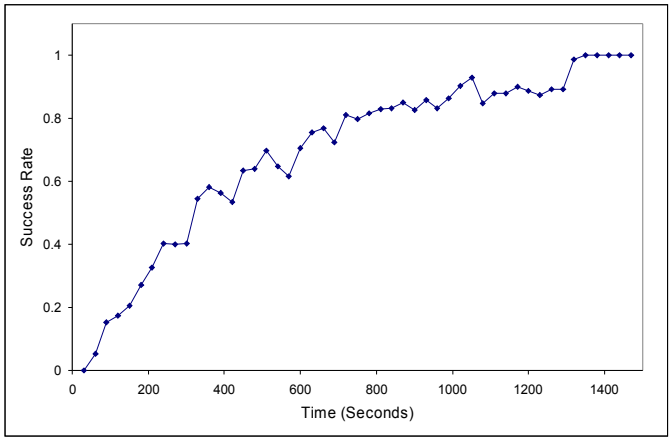

Figure 7: Rehabilitation Example, Histogram Average, Bucket Size = 30 secs

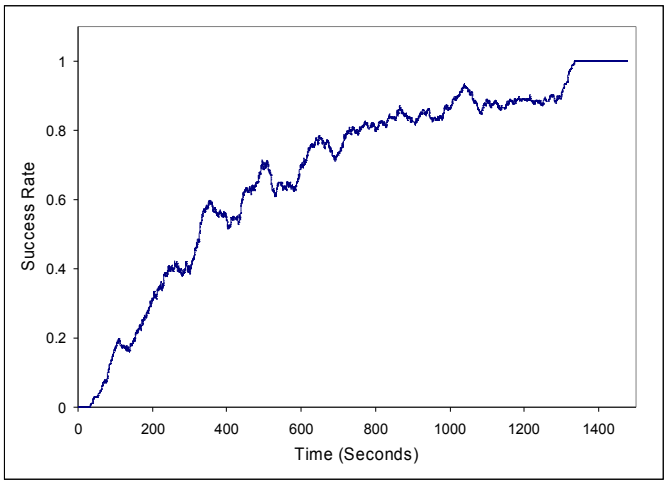

Figure 8: Rehabilitation Example, Moving Window Average, Window Size = 30 secs

4) Experiment 3 Results

E-mail rate ~ 42 msg/sec.

MetricDecayFactor = 2.5 mins.

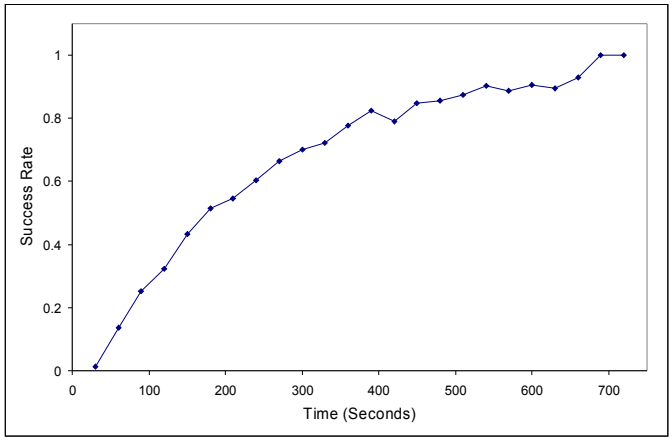

Figure 9: Rehabilitation Example, Histogram Average, Bucket Size = 30 secs

E. Conclusion:

The rehabilitation feature of Stockade executed as expected under our test conditions.

Experiment 1 and 2 show the rehabilitation process of Stockade with a MetricDecayFactor of 5 minutes. Both experiments are essentially testing the same features of stockade, but since Experiment 2 is sending e-mails at a greater rate, the averaging process is expected to resemble the rehabilitation function more closely than experiment 1.

In Experiment 3 the MetricDecayFactor was reduced to 2.5 minutes, and as the results show, this halved the amount of time required for a full rehabilitation. In addition we were sending e-mails (without any deliberate delays) at an average rate of 42 msg/sec, hence the calculated success rate began to more accurately resemble the probability of connections being accepted.

F. Limitations:

The traffic profile on a mail server may vary significantly to the test conditions.

V. Registering Spam During Rehabilitation

A. Introduction:

In a real mail server using Stockade, spam will be registered quite frequently, and a host might be registered as a spammer during their rehabilitation period. I will further investigate how Stockade will handle this type of situation.

B. Aim:

- To test whether a host can be re-blacklisted during its rehabilitation period and ensure that the rehabilitation process will continue as normal from thereon.

C. Method:

1) Equipment and setup

- 2 x PC's running FreeBSD 6.2 networked over a FastEthernet connection. They will be referred to as the mail client and mail server.
- The client had the following additional programs used throughout the testing:
  - sendmsg – A script written to send e-mails to the server at specified intervals using smtpclient.
  - Smtpclient 1.0.0.
- The server had the following additional programs used throughout the testing:
  - sendmail 8.13.8.
  - ipfw (as with FreeBSD 6.2 release).
  - Stockade 0.2.

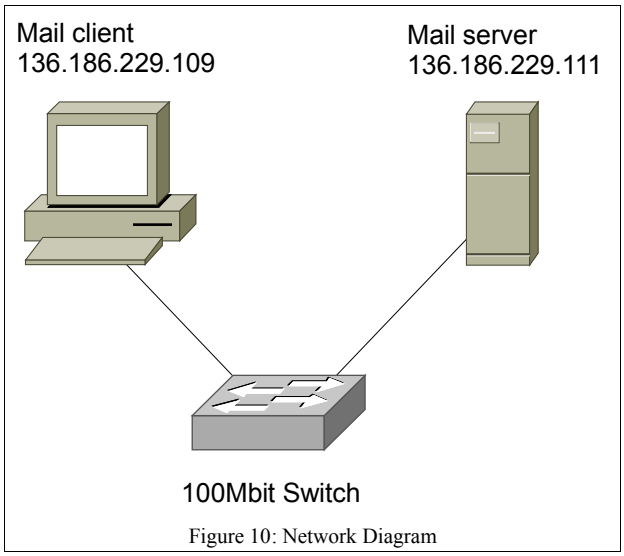

Figure 10: Network Diagram

2) Procedure – Experiment 1

- Stockade was configured with a MetricDecayFactor of 2.5 minutes.
- The client's IP address was initially given a spam rating of 1.0 by Stockade so that all incoming e-mails from the client were rejected at the start.
- E-mail messages were sent (using sendmsg) from the client to the server at 20 msg/sec.
- When the dropping probability of the client reached 0.5, the register_spam program was executed on the server to give the client a spam rating of 1.0 again.
- All successful and unsuccessful attempts were logged from the client using tcpdump.

D. Results:

1) Analysis Method

- The tcpdump file was analysed and used to determine the number of successful and unsuccessful attempts according to the guidelines below:
  - A packet with an syn and ack flag set indicated a successful connection with the SMTP server.
  - A packet with an ack and rst flag set indicated an unsuccessful connection with the SMTP server.
- A python script was used to perform a moving window average on the parsed tcpdump data.
- The results were plotted using Microsoft Excel

2) Experiment 1 Results

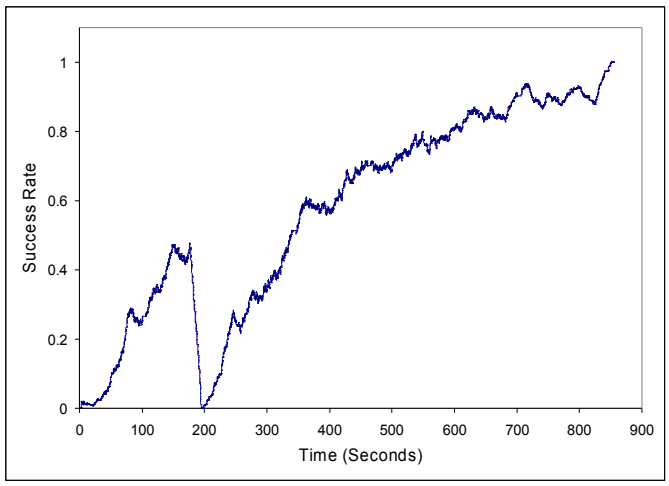

Figure 11: Spamming During Rehabilitation Example, Moving Window Average, Windows Size = 15 secs

E. Conclusion:

Registering the client as a spammer during the rehabiliation period worked as expected. Figure 11 shows the rehabilitation restarting at 100% blocking probability when the client was blacklisted the second time.

F. Limitations:

The traffic profile on a mail server may vary significantly to the test conditions.

VI. Neighbourhood Aggregation and Neighbourhood Poisoning

A. Introduction:

1) Neighbourhood Poisoning

Stockade can blacklist and greylist individual hosts as already demonstrated, however Stockade also has the ability to register multiple IP addresses as spammers with a single register_spam execution. This can be done by specifying the subnets IP address and including the number of bits in the subnet portion of the address. Here is an example of using Neighbourhood Poisoning:

register_spam Manual 192.168.0.0/24 1.0

192.168.0.0/24 will select the range of IP addresses from: 192.168.0.0 to 192.168.0.255

Stockade has the ability to register spammers on CIDR subnet boundaries as well.

2) Neighbourhood Aggregation

Neighbourhood aggregation is a feature of stockade which can reduce the number of blacklist entries by combining adjacent entries into a single entity encompassing both subnets. Stockade will merge adjacent blacklisted subnets on CIDR boundaries.

B. Aim:

To test whether Neighbourhood Aggregation and Neighbourhood Poisoning function normally.

C. Method:

1) Equipment and setup

- 1 x PC installed with FreeBSD 6.2 referred to as the mail server
- The mail server had the following additional programs iused throughout the testing:
  - ipfw (as with FreeBSD 6.2 release).
  - ipfw listing script – Lists all the entries in the ipfw tables.
  - stockade 0.2.

2) Procedure

- The Ipfw listing script was executed and left running in a console so the effects of the register_spam commands could be easily monitored.
- Various register_spam commands were typed in a seperate console.
- The results were inspected and recorded by reviewing the Ipfw tables list

The results and test procedures have both been combined in the subsection D.

D. Results:

1) Analysis Method

The results were gathered by observing the Ipfw tables list every one second.

2) Experiment 1 - Neighbourhood Aggregation Results

a) Test 1

To test neighbourhood aggregation the following commands were issued on the server running Stockade:

register_spam tag 192.168.0.0/29 1.0

register_spam tag 192.168.0.8/29 1.0

The result of executing these commands was as follows:

- A single entry was created with IP 192.168.0.0/28 and blocking probability of 1.0

b) Test 2

The second test involved aggregating two larger subnet ranges. Here is register_spam commands which were executed:

register_spam tag 192.168.0.64/27 1.0

register_spam tag 192.168.0.96/27 1.0

The result of executing these commands was as follows:

- A single entry was created with IP 192.168.0.64/26 and blocking probability of 1.0

c) Test 3

The purpose of Test 3 was to show how Stockade deals with invalid CIDR boundaries

The following was executed on the mail server:

1. register_spam tag 192.168.0.0 1.0
2. register_spam tag 192.168.0.1 1.0
3. register_spam tag 192.168.0.2 1.0

4. register_spam tag 192.168.0.3 1.0

The result was as follows:

- When the first two commands were executed, a single entity with IP address 192.168.0.0/31, and blocking probability of 1.0 was created.
- When the third register_spam command was executed an additional entry was created in the Ipfw table with IP 192.168.0.2/32 and blocking probability of 1.0
- When the fourth register_spam command was executed a single entitiy was created in the Ipfw table with IP 192.168.0.0/30 and blocking probability of 1.0

3) Experiment 2 - Neighbourhood Poisoning Results

A similar set of tests were conducted to test Neighbourhood Poisoning.

A blacklist entry was created for subnet 192.168.0.0/24 with a spam rating of 1.0.

The following commands were executed sequentially at different intervals:

1. register_spam tag 192.168.0.10 1.0
2. register_spam tag 192.168.0.50 0.5
3. register_spam tag 192.168.0.95 1.0

The results were as follows:

- The first command increased the blocking probability of 192.168.0.0/24 to 1.0
- The second command had no effect
- The third command increased the blocking probability of 192.168.0.0/24 to 1.0

E. Conclusion:

Neighbourhood Aggregation and Neighbourhood Poisoning worked as expected.

The results from Experiment 1 confirm that neighbouring blacklist entries would be combined into a single entity if they existed on valid CIDR boundaries.

Note that when blacklisting/greylisting one or more hosts which can be summarised by an existing subnet rule, the new blocking probability of the subnet inherits the value: MAX(subnet_blocking_probability, host_blocking_probability)

To simplify: the blocking probability of an entity can only be increased, and not lowered by executing register_spam.

F. Limitations:

Some of the limitations of the testing method used includes the following:

- There were minimal entries in the Ipfw tables, therefore Neighbourhood Poisoning and Neighbourhood Rehabilitation were not tested when Stockade was under a heavy load.
- No e-mails were sent from the client to the server to test Neighbourhood Poisoning more thoroughly.

VII. Limitations and Future Work

The drawbacks of the various tests have been documented under each experiments' Limitations section. I will recap those limitations and append to the list with some future work.

Some of the significant limitations of the experiments, as well as possible future work include the following:

- Stockade was only tested using the Ipfw rule-based filtering strategy. Ideally all the tests in this experiment should be repeated using socket filtering since some users may prefer this method over Ipfw.
- The mail server had a low volume of traffic to handle. Testing stockade under a heavier traffic load is recommended.
- Stockade was only tested using the RESET IpfwRejectAction. The IpfwRejectAction parameter determines how incoming packets should be disregarded. The available options are Deny, Reject and Reset. These other parameters should be tested considering they may affect how quickly a client can be accepted or denied a connection to the mail server.
- Optimal configuration settings for RehabilitationHeartBeatInterval, RehabilitationIterationsPerHeartbeat and MetricDecayFactor should be investigated. These properties can affect how regularly clients/subnets are rehabilitated.
- The Error handling of Stockade was not assessed in detail. Testing how a program handles invalid input, and displays appropriate and informative error messages is an important factor.

VIII. CONCLUSION

Section III examined Stockade's performance by comparing the bandwidth consumption with and without Stockade in use. The results concluded as follows:

In the situation where 20 clients were sending static length e-mails to a server in sequential order, and 4 of those clients were spammers, an approximate 93% reduction in spam traffic was achieved by running Stockade on the mail server, as opposed to not using Stockade at all. The results also showed that Stockade did not refuse connections to any IP addresses which weren't registered as spammers.

In Section IV we experimented with Stockade's automatic rehabilitation feature. From the resulting graphs (Figure 6 through to Figure 9) we can conclude that this feature was functioning normally. The blocking probability during rehabilitation follows an exponential curve until the minimum cut off is reached, after which the client is removed from the greylist. Halving the

MetricDecayFactor also had the desired effect of reducing how quickly a client would be rehabilitated.

Section V considers the scenario where a client sends a spam e-mail, and then repeatedly sends ham e-mails for some time before sending another spam message. The results showed that in this scenario Stockade was able to restart the rehabilitation from 100% blocking probability when the client sent the second lot of spam. After the client was blacklisted for the second time the rehabilitation progressed normally until the client was entirely removed from the greylist.

Section VI experimented with Neighbourhood Aggregation and Poisoning the Neighbourhood. The test cases concluded that adjacent neighbourhoods which existed on valid CIDR boundaries were able to be merged into a single rule encompassing both subnet ranges. This can reduce the number of blacklisted and greylisted entries, hence reducing the mail server's overhead.

References

[1] M. Robb, G. Armitage, Stockade, http://caia.swin.edu.au/stockade/ (viewed 2 April 2007)

[2] "The FreeBSD Project", http://www.freebsd.org (viewed 2 April 2007)

[3] FreeBSD Manual, http://www.freebsd.org/cgi/man.cgi (viewed 2 April 2007)

[4] "FreeBSD Handbook Chapter 27 Firewalls", http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html (viewed 2 April 2007)

[5] "The FreeBSD Ports Archive", http://www.freebsdsoftware.org/mail/smtpclient.html (viewed 2 April 2007)

[6] SpamAssassin, http://spamassassin.apache.org (viewed 2 April 2007)

[7] SpamBayes, http://spambayes.sourceforge.net (viewed 2 April 2007)

[8] Procmail, http://www.procmail.org (viewed 2 April 2007)

[9] M. Tran , "Mitigating Email Spam by Statistical Rejection of TCP Connections Using Recent Sender History", Australian Telecommunication Networks and Application Conference 2006, Dec 2006 (http://caia.swin.edu.au/pubs/ATNAC06/Tran_M2m.pdf) (viewed 2 April 2007)