Distributing Network Identity to Mitigate Denial-of-Service Attacks

by

Pallavi Naresh

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

© Massachusetts Institute of Technology 2005. All rights reserved.

Author .................
Department of Electrical Engineering and Computer Science

August 15, 2005

Certified by ............
William Weinstein
Principal Member of the Technical Staff
Thesis Supervisor

Certified by ..................
Howie Shrobe
Principal Research Scientist
Thesis Supervisor

Accepted by ..............
Arthur C. Smith
Chairman, Department Committee on Graduate Students
Distributing Network Identity to Mitigate Denial-of-Service Attacks

by

Pallavi Naresh

Submitted to the Department of Electrical Engineering and Computer Science on August 15, 2005, in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

Abstract

The CONTRA, Camouflage of Network Traffic to Resist Attacks, project was carried out by Draper Laboratory to provide a defense mechanism against distributed denial of service (DDoS) attacks to both prevent DDoS attacks and mitigate their effects. This Masters project looks at the CONTRA system and assesses its effectiveness. The goal of this project is to explore whether the techniques employed by CONTRA—namely IP dispersion, redundancy, and traffic masking, can effectively mitigate the effects of a DDoS attack. The analysis provides a set of recommendations for operating the CONTRA system to impede an outside attacker.

Thesis Supervisor: William Weinstein
Title: Principal Member of the Technical Staff

Thesis Supervisor: Howie Shrobe
Title: Principal Research Scientist
Acknowledgments

This thesis was prepared at The Charles Stark Draper Laboratory, Inc.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

I would like to give thanks to my research supervisor, William Weinstein, for his support and guidance. This thesis would not have been possible without the time, resources, and inspiration he has given me.

I would also like to thank the Charles Stark Draper Education Office, and Anne Hunter of the MIT EECS Undergraduate Office.

Finally, I would like to acknowledge my family: my parents, Naresh Chand and Rekha Naresh, and my siblings, Ravi and Himani Naresh. I thank them for their love and support, and all the opportunities they have provided me with.
ASSIGNMENT

Draper Laboratory Report Number T-1535.

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.

______________________________  ______________________________

_____  ______________________
Contents

1 Introduction 15
   1.1 Motivation ........................................... 15
   1.2 Organization ........................................ 16

2 An Overview of the CONTRA System 19
   2.1 Overview of CONTRA Techniques ......................... 19
   2.2 IP Dispersion, Modifying Address Bindings, and Infoservers .... 21
   2.3 Redundancy and K-of-N Encoding ........................ 22
   2.4 Masking Traffic and Packet Information: Relays and Encryption . 22

3 The Attacker Problem 25
   3.1 Purpose ................................................ 25
   3.2 An analogy ............................................ 25
   3.3 Problem Overview ...................................... 26
   3.4 Attacker: Assumptions, Strategy, and Parameters ............. 27
      3.4.1 Assumptions ..................................... 27
      3.4.2 Strategy ......................................... 28
      3.4.3 Parameters ...................................... 30
   3.5 CONTRA: Defender Assumptions, Strategy, and Parameters ..... 31
      3.5.1 Assumptions ..................................... 31
      3.5.2 Strategies ....................................... 32
      3.5.3 Parameters ...................................... 33
4 Simulation Design: Model Overview and Implementation

4.1 Overview and Objectives
4.2 Model Components
4.3 Implementation
4.4 Client Model
4.5 Simulating Network Traffic
   4.5.1 Client Traffic Generator Module
   4.5.2 CONTRA Relayer
4.6 The Infoserver
4.7 Servers Module
4.8 Simulating an Attack
4.9 Summary of Simulation Parameters

5 Methods and Results

5.1 Data Collection
5.2 Attacker Strategy: Producing Correlations
5.3 Attacker Probabilities
5.4 Results
   5.4.1 Default Parameters
   5.4.2 Attacker's Analytical Time Frame
   5.4.3 Traffic Distributions
   5.4.4 Addresses to Address Space Ratio
   5.4.5 K-of-N Encoding
   5.4.6 Dummy Traffic

6 Conclusion

A CONTRA Simulation User Manual

A.1 Overview
A.2 Topology Files
A.3 Message Definitions and Communication Channels
A.4 Modules .................................................. 72
A.5 Event Logging .................................................. 73
A.6 Configuration and Parameters ................................................. 74
A.7 Compiling and Running .................................................. 75

B Explanation of Combinatorics .................................................. 77
B.1 Problem Statement .................................................. 77
B.2 Solution A .................................................. 78
B.3 Solution B .................................................. 79
List of Figures

3-1 Attacker Scenario .......................... 29
4-1 Topology of the CONTRA simulation .......................... 38
5-1 Variations in C, for t = 2 and k = 2 .......................... 50
5-2 Attacker Probabilities with Variations in t .......................... 51
5-3 Attacker Probabilities with Variations in k .......................... 53
5-4 Attacker Probabilities with Variations in Traffic Distributions .......................... 57
5-5 Attacker Probabilities with Variations in Relay Delays .......................... 58
5-6 Attacker Probabilities with Variations in Addresses Per Server .......................... 61
5-7 Attacker Probabilities with Variations in Number of Servers .......................... 62
5-8 Attacker Probabilities with Variations in N .......................... 64
5-9 Attacker Probabilities with Variations in Dummy Traffic Percentages .......................... 66
List of Tables

4.1 A list and description of all the available simulation parameters . . . 44

5.1 Summary of default parameter values . . . . . . . . . . . . . . . . . 49
Chapter 1

Introduction

1.1 Motivation

In a distributed denial of service attack (DDoS), an attacker inundates a network with spurious requests in order consume critical resources and degrade or disrupt a service. Network attackers launch DDoS against individuals, businesses, and political organizations. A CAIDA/UCSD 2001 study estimated more than 12,000 DDoS attacks worldwide against at least 5,000 distinct hosts in just a 3 week period [4].

In order to combat this serious threat to Internet hosts, we must develop both preventative and responsive techniques and analyze their effectiveness. Current technologies focus largely on responsive measures. The CONTRA, Camouflage of Network Traffic to Resist Attacks, project was carried out by Draper Laboratory to provide a defense mechanism against DDoS attacks to both prevent DDoS attacks and mitigate their effects. This Masters project looks at the CONTRA system and assesses its effectiveness.

This project has relevance to two critical areas of defense that are particularly significant to this day and age. The first motivation for undertaking this specific project is the same motivation behind that of the CONTRA project as a whole. That is to prevent and respond to DDoS attacks to ensure a network’s integrity. DDoS attacks pose a serious threat to businesses on a day to day basis. A DDoS attack is relatively easy to mount and there are a multitude of freely available tools to allow
attackers to disrupt critical services. CONTRA seeks to provide a defense against DDoS attacks. An important component in developing a method of defense involves understanding possible attack scenarios, which is what this project seeks to do. The goal is to determine whether an attacker with a given set of resources can mount a successful attack against the CONTRA system and whether there are additional measures CONTRA can take to make such an attack more difficult to mount.

Apart from addressing the direct needs of CONTRA, this project also attempts to understand how to correlate seemingly disjoint events from a large data set to pinpoint a specific phenomenon. In the case of CONTRA, these events involve network packets and IP addresses. But in a broader context, the events could be anything from the purchase of an airline ticket to a suspicious phone call. And so while this project seeks to launch an attack on the CONTRA system, the methods and analysis can also be used as a defensive tool in, for example, a national security system.

According to National Defense Magazine, "‘information fusion’ [is the] key to winning wars.[3]" US Defense organizations that have not been willing to share their information with other systems are now attempting to make their information available to other organizations on a plug-and-play "global information grid." Such an information grid could benefit from an event correlation tool. Thus this project also holds implications for global defense work that seeks to monitor and correlate the work of suspected adversaries while securing secrecy.

1.2 Organization

The outline of this paper is as follows: Chapter 2 describes the CONTRA system, beginning with a high level overview and then delving into the system’s structural details. Chapter 3 further explains the aim of this study and formalizes the problem of accessing the effectiveness of CONTRA from an attacker’s perspective. Chapter 4 provides the design of the simulation used to carry out this study. It describes how the CONTRA system was abstracted to create a discrete event simulation for the purposes of analysis. Chapter 5 presents the methods of data collection and
the resulting analysis. Chapter 6 provides a list this work’s contributions as a final conclusion.
Chapter 2
An Overview of the CONTRA System

This section provides a high level overview of the CONTRA system, while ignoring the implementation details. Additional information can be found in the CONTRA final report [1]. The aim of this project is to model the structure of the CONTRA system and analyze its behavior. Providing an overview of the techniques employed by the CONTRA system is thus sufficient for the purposes of this project. The CONTRA system, developed at Draper Laboratory, uses network traffic masking techniques to mitigate the effect of a DDoS against critical members of a user community on the Internet. A DDoS attack attempts to direct sufficient “noise” packets to a server’s IP address to overwhelm the server’s ability to handle legitimate packets. A CONTRA system comprises a set of cooperating hosts that communicate among themselves over the Internet. These hosts mask their identity and their traffic patterns so that if an attacker had the resources to launch a DDoS attack, he would not know where to direct those resources in successful manner.

2.1 Overview of CONTRA Techniques

The following outlines the key techniques the CONTRA System employs in order to achieve its objectives.
1. IP Dispersion and Modifiable Address Bindings.

- Each CONTRA server is characterized by multiple IP addresses on multiple hardware ports. IP addresses are chosen randomly from a large pool of addresses and bindings are kept secret from non-CONTRA hosts.
- Each server is equipped with a mechanism to detect a traffic imbalance across its ports. If it detects an imbalance, the server will drop traffic on the flooded port before the traffic reaches its communication stack.
- Servers can change their addresses randomly or in response to an attack.


- Each CONTRA host communicating with another CONTRA server adds redundancy to its message. It then breaks the message up into $N$ segments and sends each segment to a different IP address belonging to the destination server.
- A CONTRA server requires only a subset, $K$, of the $N$ segments in order to reconstruct the message.
- If a server must drop a port or change an address, it can reconstruct its messages from the traffic to the remaining ports.


- A message is relayed through one or more different CONTRA hosts, each acting as a Chaum Mix, to impede traffic analysis.
- The IP addresses of the source and final destination hosts, and the message contents are encrypted with the key of the next relay host.
- For any packet, an eavesdropper can only see the destination IP addresses of individual hops.

4. Specialized DNS Servers: Infoservers
An Infoserver is a designated CONTRA server that provides bindings between CONTRA hosts, IP addresses, and encryption keys.

An Infoserver provides a list of relays to any CONTRA host attempting to communicate with another host.

Communication between CONTRA hosts and Infoservers are authenticated and encrypted.

In the following sections, I discuss each of these aspects of the CONTRA system in more detail.

2.2 IP Dispersion, Modifying Address Bindings, and Infoservers

Every CONTRA host has multiple IP addresses bound to multiple physical hardware ports. A host can detect a traffic imbalance over its ports and drop traffic on an inundated port before the traffic reaches its communication stack. All messages sent to CONTRA hosts have added redundancy and are broken up and sent to multiple IP addresses on the destination host. Thus even if a host drops an address, it is able to recover legitimate traffic over its other addresses.

As such, an attacker would need to correlate multiple addresses with a single server in order to successfully disrupt communication to a single host. In order to launch a successful attack against a CONTRA host, an attacker would have to flood multiple addresses associated with a single host. The host and address bindings are not publicly available. Thus an attacker's only resource is to correlate addresses by watching network traffic.

Patterns in the network traffic are masked and all critical hosts in a CONTRA system are able to protect themselves by changing their address over time. The specialized DNS servers, called the Infoservers, that provide host to address bindings are also considered critical hosts. They too can change their addresses when appropriate. An ordinary CONTRA host needs to contact an Infoserver in order to find other hosts.
CONTRA provides a mechanism for hosts to locate and maintain knowledge of at least one Infoserver. This mechanism also allows the whole system of CONTRA hosts to sync up initially. Since a CONTRA host has full access to all the name bindings in CONTRA, via an Infoserver, a compromise of a CONTRA host would provide a way for an attacker to easily learn the addresses of all CONTRA hosts. Thus CONTRA employs a hardware based encryption mechanism, similar as a smart card, to encrypt all keys into and out of a host. This prevents a network-based attacker from learning critical information about the CONTRA system.

2.3 Redundancy and K-of-N Encoding

Each message sent across the CONTRA system is encoded with redundancy and transformed into multiple packets. It is then transmitted over distinct paths and reassembled by the receiving host. Each message is broken into $N$ blocks in such a way that the message can be reconstructed from a subset $K$ of the $N$ blocks. The implementation details of this transformation are not necessary for the purposes of this project, but algorithms to do so are readily available in computer science literature [5]. As described earlier, each block arrives on a unique address on the receiving host. $K$-of-$N$ encoding allows a host to drop a finite number of addresses and still reconstruct legitimate incoming messages.

2.4 Masking Traffic and Packet Information: Relays and Encryption

An important function of the CONTRA system is to mask network traffic patterns among CONTRA hosts. An attacker launching a DDoS attack can be successful if he correlates a set of addresses as belonging to a particular host. Thus the network traffic should not yield information that will reveal correlations between network addresses.

In order to obfuscate the network traffic, CONTRA packets are not sent directly from a CONTRA source to a CONTRA destination—they are always relayed by at
least one other CONTRA host. All CONTRA hosts act as relays for other hosts. Each packet output by a CONTRA source is directed to a different relay host (with respect to that source) until the available set of relays has been exhausted. Then the relay pool can be reused.

The relays prevent an attacker from associating an observed packet with both the CONTRA source host and the CONTRA destination host. The CONTRA relays implement a mechanism known as a Chaum Mix [2]. The purpose of such a relay is to prevent an observer of traffic from correlating inbound packets with outbound packets. In order to accomplish this, the following characteristics of inbound packets are also modified before they are retransmitted:

- Contents. Packets are encrypted using the encryption key of the next relay in the transmission.
- Length. Random sized padding is added to each packet.
- Timing. Packets are not retransmitted in the order that they arrive.

A minimum level of traffic is necessary in order for a mix to work effectively. When a source generates a CONTRA packet it can specify how many relays the packet must traverse before being delivered to the destination. There must be at least one relay, and that first relay is the only one selected by the source host. Subsequent relays are selected by the relay hosts.

When a packet arrives at a CONTRA host, the host first decrypts the payload and verifies that it is a valid CONTRA packet. Then it checks a parameter to see how the packet should be relayed or if it has reached its final destination. If the parameter is > 1 then the packet must be sent to another relay host. The relay decrements the parameter, re-encrypts the payload with the key of the next relay, and transmits the packet. If the parameter equals one then the relay strips of the outer part of the CONTRA header. What remains has already been encrypted for the actual destination. The relay then transmits it to the final destination.
Chapter 3

The Attacker Problem

3.1 Purpose

The purpose of the CONTRA system is to allow a network to mitigate the effects of DDoS attacks. CONTRA attempts to provide protection against DDoS attacks using the techniques discussed in the previous section. However a complex system such as CONTRA is only as secure as careful analysis proves it to be. Even so, over time weaknesses in the system and new analytical techniques may be discovered. It is still important to assess the system from the viewpoint of the attacker to understand the dynamics of the system and comprehend what weaknesses exist and what, if any, further protection mechanisms may be adopted. In this chapter, I present the attacker problem and the CONTRA perspective.

3.2 An analogy

Conceptually, the problem at hand is similar to the children's game of battleship where two players are given a grid on which they must place their "battleships." The players cannot see one another's grids but must randomly list points to hit on their opponent's grid in hopes of finding and sinking their opponent's battleships. If a player's battleship is hit, he announces it has been hit and if the battleship has been hit on all the grid points on which it spans, he announces it has been sunk.
We can define our game between two players: the CONTRA system (the defender), and the attacker. CONTRA’s “battleships” are the servers it is trying to protect and the playing “grid” is the address space the servers are hidden in. The locations that define the servers are addresses within the address space and CONTRA may move the servers to new addresses at any time. The attacker’s objective is to “sink” CONTRA’s servers by determining the location of the servers and bombarding them with requests.

The dynamics between CONTRA and the attacker become more complicated as we introduce the various parameters of the system. However the analogy serves to elucidate the problem. The notion of “winning” or losing the “game” is unclear. The goal ultimately is to understand the system characteristics and behavior to determine whether either player, the attacker or CONTRA, can adopt a strategy to ensure success within reason. Additionally, the players are further constrained in that they both have limited resources.

3.3 Problem Overview

An attacker can not use a traditional approach in order to successfully launch a DDoS attack against CONTRA. As discussed earlier, by employing IP dispersion and $K$-of-$N$ encoding, the CONTRA system is resistant to flooding attacks against any particular IP address. In order to achieve any measure of success, an attacker must flood multiple addresses of a single server—at least $N - K + 1$ of them. Certain assumptions must be made regarding the attacker’s resources. For example, if he attempts to flood too many addresses at once the impact of the attack will be diluted. The attacker’s best hope is to correlate addresses belonging to a single host and to flood just those addresses. After a host has been attacked, it changes its address. In order for the attacker to inflict a sustained flood against a given host, he must discover the new set of IP addresses belonging to that host, and then reinstate the flood. The resulting “game” between the attacker and CONTRA can be described as a pair of interacting systems each of which is characterized by many parameters that can be
observed and controlled. The attacker and CONTRA interact via those parameters they have in common. While there may be other vulnerabilities and possible modes of attack beyond DDoS attacks on the CONTRA system, I focus solely on how an attacker may watch network traffic and correlate addresses in order to increase his chances of success. I evaluate how well CONTRA strategies work in masking traffic patterns and server addresses by performing analysis on behalf of a possible attacker.

### 3.4 Attacker: Assumptions, Strategy, and Parameters

#### 3.4.1 Assumptions

In order to analyze possible modes of attack, some assumptions must be made about the attacker’s resources and capabilities. The following explains these assumptions:

1. CONTRA is intended to mitigate flooding attacks by a strong attacker—a nation state or a well-funded extra-national organization. Thus the attacker may have significant financial and political resources to invest in technology to aid in launching a DDoS attack.

2. The attacker is aware of the existence of CONTRA systems and understands how such systems operate. The attacker may obtain such information from this study as well as other related, unclassified papers. It is not practical to try and keep this information secret and in the security community an open architecture is generally considered more secure than an architecture that relies on secrecy.

3. The attacker can learn what blocks of IP addresses are assigned to possible host targets of interest. The attacker cannot however, determine a priori, which particular IP addresses belong to a specific host. The success of the CONTRA system relies on keeping this information secret, and CONTRA protects this information with the various mechanisms previously discussed.
4. The attacker may be able to compromise a particular CONTRA host via an
attack over the Internet. However, as discussed earlier, the CONTRA proxy
operations are embedded in hardware and critical CONTRA information will
never be available to the attacker in unencrypted form.

5. Although a strong attacker is assumed, the attacker does not have the resources
to launch a successful DDoS attack across all observable IP addresses. The
attacker must direct his resources carefully.

6. By assumption 1, the attacker has the ability to observe traffic on any link of
the Internet. This includes “backbone” links and “last mile” links between ISPs
and user facilities. The attacker can sniff anywhere, and perhaps everywhere si-
multaneously, on the Internet, but because each packet comprising a particular
message is relayed through a different CONTRA host he cannot trace the pack-
ets through the network from CONTRA source host to CONTRA destination
host.

7. Although by assumption 1 the attacker is well funded, he cannot decrypt any
encrypted CONTRA information at a significant rate to allow for an attack
based on the decrypted information. CONTRA employs strong encryption.

3.4.2 Strategy

By assumption 3, an attacker has no knowledge beforehand of which IP addresses
within a particular IP space belong to critical CONTRA hosts. His strategy must
then be to observe network traffic and somehow correlate IP addresses with hosts.
The best place for the attacker to listen is at a pipe leading into a facility containing
critical hosts. (See Figure 3-1.) For example, the attacker might look for a pipe going
into a military base or a corporate server farm. Inbound traffic to any CONTRA host
inside the facility at this point in the network will be visible. Even if the traffic is
split among multiple pipes entering the facility from different ISPs, the traffic may
be modeled as entering from one pipe since the attacker can monitor all pipes at the
same time. The farther upstream that an attacker attempts to monitor traffic, the less likely he is to see all of the packets destined for hosts in the target facility. Thus, I assume the attacker is watching traffic at a point directly leading to a host facility.

The attacker will only find inbound traffic characteristics useful. Monitoring outbound traffic from a facility provides little information because the source addresses in the packet headers are all spoofed and the destination address only reveals the address of the first hop.

Assuming that the attacker is able to gather enough information to understand where he should direct his resources, the attacker must address two issues when mounting a large scale flooding attack. The first is generating a sufficiently high volume of flooding packets, and the second is not getting caught. Generating the flood from his own machine is bad strategy for the attacker. A single computer cannot produce enough traffic to have a significant impact on a target, and a direct attack is easy to trace back to its source.

The attacker could instead launch a distributed attack by compromising a large number of vulnerable hosts on the Internet to use them to generate the flooding traffic. The attacker can further insulate himself by having the compromised hosts reflect their attack off of other Internet hosts. If the attacker retains no control channel

Figure 3-1: Attacker Scenario
to the compromised hosts then the targeted CONTRA host can move out from under
the attack by simply changing its addresses one time. The more interesting case is
when the attacker has the ability to control the compromised hosts in real time and
can cause them to flood new IP addresses. I assume the attacker has the capability
to do so, resulting in an interactive attacker response scenario.

I also assume that the attacker does not have the ability to flood the entire in-
coming bandwidth into the facility. If the attacker could do so, then of course any
further analysis would be inconsequential. I make no assumption on the size of the
incoming bandwidth.

As the attacker launches multiple attacks, he begins to reveal information about
himself and CONTRA administrators or defense personnel may be able to determine
the location of the attacker. I do not make any assumptions on how many chances
the attacker has to launch an attack before he is effectively terminated. Such an
assumption would be based on the resources available to CONTRA administrators,
which is not the focus of this study.

3.4.3 Parameters

The parameters of interest in the attacker problem are as follows. These parameters
will vary on the CONTRA parameters of defense, which I discuss in the next section.

1. Packet Characteristics. Because the payload in a CONTRA packet is encrypted
the only parameters of an inbound packet observable to an attacker are:

   • time of arrival
   • destination IP address
   • packet length

2. Time to identify a host. Can the attacker correlate addresses with hosts by
watching inbound traffic? If so, how long does the attacker need to monitor
inbound traffic in order to correlate CONTRA hosts and addresses? In order
to affect just one host, he must identify $N - K + 1$ addresses as belonging to a single host.

3. Time to identify a host again. A targeted CONTRA host can change its addresses and move out from under an attack. Can the attacker find a host he has already attacked by watching inbound traffic? If so, how long does he need to do so?

### 3.5 CONTRA: Defender Assumptions, Strategy, and Parameters

#### 3.5.1 Assumptions

In order to analyze the attacker problem, some assumptions must also be made about how the CONTRA system behaves and its ability to defend itself. The following lists my assumptions.

1. The CONTRA system can successfully recover from a single flooding attack. CONTRA can detect a traffic imbalance and thus a flood against an IP address. When it recognizes an attack, it replaces the affected addresses from a pool of spares. If enough addresses on a given server are affected, it may even change its entire set of addresses.

2. A sustained attack against a particular server involves multiple attacks to that server. The attacker must continually reanalyze the traffic to detect that the attack target has changed addresses and infer the new set of addresses for that specific targeted server.

3. Traffic composition. If the probability of occurrence of inbound IP addresses were uniformly distributed across the entire potential destination addresses within a facility, and if that distribution remained constant over time, then no information could be inferred by the attacker. Such traffic is statistically “white”. I assume traffic to CONTRA servers is NOT statistically white.
I do not strictly define the notions of “winning” and “losing” the attacker game. If the attacker can correlate addresses to find a particular server, the attacker achieves a temporary win. If attacker can find a particular server again, after it has changed addresses, the attacker achieves a more significant win. However, no assumptions are made about how long it may take for a server to detect an attack and how many attacks a particular server can sustain. I am interested in how long it takes for the attacker to analyze the traffic and gain something useful. The ability for CONTRA to sustain multiple attacks is based on practical hardware and performance characteristics of the system, the analysis of which are beyond the scope of this study.

3.5.2 Strategies

The primary strategy for CONTRA, as I have discussed before, is to move out from under an attack once an attack has been detected by changing the addresses of an affected server. In assumption 3 above, I state that the inbound traffic into the facility housing multiple, critical hosts is not statistically white. Thus the attacker should be able to gain information by observing this traffic. After understanding what information is revealed to an attacker through traffic statistics, CONTRA can attempt to impede the attacker’s ability to correlate actual IP addresses to CONTRA servers.

CONTRA is able to observe its own inbound traffic into a facility, just as the attacker can. CONTRA, however, already knows the relationships of IP addresses to servers and can use this knowledge for generating dummy traffic to shape the statistics. The dummy traffic generator must sniff inbound traffic in order to determine its traffic generation strategy. The dummy traffic it generates must be sent through a relay outside of the facility in order to appear on the inbound traffic flow. Furthermore, it needs to be sent through a collection of relays just as normal CONTRA traffic is. Otherwise, by monitoring relay traffic, the attacker may detect a bias towards a particular destination to infer information about the volume of incoming dummy traffic.

I assume that CONTRA has the ability to generate dummy traffic in such a
convincible manner. Producing dummy traffic may prevent the attacker from drawing inferences altogether or simply control the time required by the attacker to draw such inferences. After an initial attack, CONTRA can use dummy traffic in order to obstruct the attacker's ability remount an attack. At a certain point, the attacker's cycle may become sufficiently long such that the attack is effectively diluted. It is important to note that as the attacker's attack rate increases his effectiveness may increase, but his risk of detection also increases. CONTRA may also modify other parameters of legitimate inbound traffic in order to shape inbound statistics. I discuss the parameters available to CONTRA in the next section.

3.5.3 Parameters

In order to shape incoming traffic characteristics, CONTRA has several parameters at its disposal. The following provides an outline of the parameters of interest:

1. $N$ and $K$. Each message is encoded such that it is broken up into $N$ pieces, $K$ of which are needed for the end recipient to reconstruct the message. (See section 2.3.) CONTRA may set $N$ and $K$ system wide or vary the parameters per host or even per message.

2. Addresses per server. Each server is assigned a set of addresses. CONTRA may vary the number of addresses available to each server or the algorithm for assigning addresses to servers.

3. Address Space. The set of CONTRA hosts are essentially hiding in a large address space. CONTRA may vary the size of the address space to affect the ratio of addresses in use to total addresses available.

4. Packet padding. Each packet receives some amount of dummy padding in order to whiten the incoming traffic and impede address correlation. CONTRA may vary this padding to affect incoming statistics.

5. Rate of Address Change. CONTRA may periodically change the addresses
assigned to a particular host. The rate at which it does so may affect the attacker's ability to analyze traffic.

6. Dummy Traffic. When generating dummy traffic, CONTRA has several important questions to address. What addresses should the dummy traffic be directed at? When and how often should dummy traffic be initiated? How much dummy traffic should be created?

The optimal setting of the above parameters will depend on the attacker's ability to analyze incoming traffic.
Chapter 4

Simulation Design: Model
Overview and Implementation

In this chapter, I present the design of the simulation I created to model the CONTRA system and conduct analysis. I compare the simulation model to the actual CONTRA system and discuss the model's extensibility as well as its simplifications and limitations.

4.1 Overview and Objectives

In section 3.4.2, I introduced the attacker scenario. The aim of the simulation is to provide a mechanism to study the behavior of the CONTRA system under this scenario. The simulation should produce a realistic picture of what an outside attacker of the system would see, a method for conducting analysis on behalf of the attacker, and a mechanism to simulate and study attacker interactions with the system. The objective of the simulation is not to actually implement the CONTRA system. Rather, the goal is to make certain assumptions regarding the system's behavior, and understand how an attacker could infer critical information under such assumptions.

Extensibility and flexibility are critical requirements for the CONTRA simulation. As I describe in the following sections, some implementation details and specific characteristics of the system are unknown. Thus the simulation should be modular,
in order to allow substitution of different models of behavior into the system. Each module should be parameterized to allow the simulation to easily run under a broad range of conditions. Additionally, the simulation should allow for an attack on the system and allow measurement of the system's resulting integrity.

4.2 Model Components

The scenario the simulation models is described in figure 3-1. The attacker views traffic entering a facility containing a set of critical CONTRA hosts. Incoming traffic originates from a set of CONTRA hosts outside the facility. I refer to these hosts as the clients and the hosts inside the facility as the servers.

The simulation does not try to provide an implementation of the CONTRA system, but rather it tries to provide an accurate depiction of certain important observable and measurable system characteristics. The simulation models the following aspects of the CONTRA system:

- **Client Traffic.** An attacker may observe incoming messages between clients and servers. The simulation models only the following observable message characteristics:
  - time of arrival
  - packet length
  - destination IP

- **Servers.** The CONTRA servers inside the facility receive messages and decode them using $K$-of-$N$ decoding. Since the actual packet contents are not modeled, the servers do not need to actually perform any decoding. They must simply keep track of incoming packets to provide statistics as to how many messages would actually have been decoded given a similar real scenario. Servers also receive attacks and respond to attacks by changing their addresses.
• **Dummy Traffic Generator.** In section 3.5.2, I describe how CONTRA may generate dummy traffic in order to whiten traffic characteristics and confuse the attacker. The Dummy Traffic Generator generates spurious traffic to CONTRA hosts or unused addresses in the server space with the same observable packet characteristics as regular client traffic.

• **InfoServer.** The Infoserver maps each server to $M$ addresses. The Infoserver provides this mapping to clients as well as a mechanism for servers to change addresses.

• **Attacker.** The attacker performs the analysis on the client traffic. The attacker also launches attacks against the system.

The following sections describe the framework used to implement the simulation and provide an in-depth description at each module in the simulation.

### 4.3 Implementation

To implement the simulation, I use OMNeT++, an object-oriented, public source discrete event simulator. I chose OMNeT++ because of the ease of which it can create modular, flexible network simulations. OMNeT++ is available free for academic use and has a large developer community which provides public support for the framework. Information about OMNeT++ can be found at [http://www.omnetpp.org/](http://www.omnetpp.org/).

An OMNeT++ simulation consists of modules connected together by channels to form a network. Modules communicate with other modules by sending messages through these channels. Figure 4-1 depicts the modules in the CONTRA simulation. Messages originating at the Client Module represent network traffic through the system. Each message has a unique time stamp and is eventually relayed to the Servers Module, which records its receipt.

To conduct the attacker analysis, I use Matlab as it readily provides many useful numerical analysis tools. OMNeT++ records events in what it terms as “Vector Files.” From the event logs, I use Matlab to parse and analyze the date. The tools
Figure 4-1: Topology of the CONTRA simulation
used to perform the analysis are described in the next chapter. For more detailed information about the implementation of the simulation refer to Appendix A.

4.4 Client Model

In the attacker scenario laid out above, the CONTRA system consists of a set of a trusted clients communicating with a set of trusted servers. For the purposes of analyzing an attacker's viewpoint, I am only interested in modeling client network traffic and the integrity of the traffic as it travels through the network. Rather than model clients individually, I concentrate on generating network traffic as a parameter of how many clients exist. Thus clients are effectively modeled as simply a number. The model for generating network traffic uses this number and also makes certain assumptions about the nature of the clients.

4.5 Simulating Network Traffic

Although an attacker would potentially see both incoming and outgoing traffic in the facility, I simulate incoming traffic only. As discussed in section 3.4.2, outgoing traffic does not give the attacker information about how the addresses of the CONTRA servers are correlated.

The functionality to create client traffic is broken up into two modules. The Client Traffic Generator schedules messages to be sent on behalf of clients to the servers. The Contra Relayer sends these messages to the Servers module according to the CONTRA protocol.

4.5.1 Client Traffic Generator Module

The Client Traffic Generator generates and schedules messages between clients and servers. The messages have the following characteristics:

1. Source Client
2. Destination Server

3. Time of Generation

Although the attacker will not be able to determine the source client, a source client (identified by a number) is specified for the purposes of generating realistic traffic. Traffic is generated according to the following algorithm:

- Traffic to each server is modeled as a Poisson process. The rate of message arrival to each server is chosen at random from a specified lower bound and upper bound.

- The Traffic Generator Module generates messages according to the Poisson distribution assigned to a particular server.

- After the Traffic Generator generates a message, it sends it to the Relayer Module. For each message received, the Relayer Module chooses a time delay from an upper and lower bound. It then sends the message according the CONTRA protocol to the Servers Module.

- The distribution rate of traffic to any given server may change periodically. The rate of change per server can either be set to zero or chosen randomly from a pluggable distribution.

A burst of traffic to a server occurs when the lower and upper inter-communication time bounds are randomly chosen as small values. A lack of frequent communication occurs when the time bounds are large values. The nature of internet traffic and models for simulation is a topic of current research and the true behavior of network traffic is debatable. However, I make reasonable assumptions about the nature of client-server traffic. The design of the simulation is such that if a more reasonable or accurate model for traffic generation becomes available, it may be substituted for the current traffic generator module.
4.5.2 CONTRA Relayer

The CONTRA Relayer is responsible for relaying traffic according to the CONTRA protocol. For each original message it receives from the Traffic Generator module, the Relayer generates $N$ message blocks representing the CONTRA encoding of the original message. It provides the destination server number to the Infoserver and in return receives $N$ of $M$ addresses belonging to that server. For each block, it chooses a random delay time from a pluggable distribution to represent the delay produced by the CONTRA relays. These messages travel to the Servers module and contain the following fields:

- Time of Generation
- Time delay
- Length
- Destination server
- Destination address
- Message ID

4.6 The Infoserver

A real CONTRA system would have multiple Infoservers. I simulate just one Infoserver, which is sufficient for the purposes of this study. When the simulation runs, the Infoserver module is the first module to initialize. It assigns servers to a set of random addresses. Once it is initialized, it can respond accordingly to the following request messages:

- request for a set of address bindings for a particular server
- request for the ID of the server belonging to a particular address
- request to reset all addresses for a particular server
• request to reset all addresses for all servers

4.7 Servers Module

The job of actual CONTRA servers is to receive messages and respond to them. Receiving messages involves reconstructing packets received over multiple ports. As I discussed earlier, I am not interested in simulating responses to client requests. I assume there is no information revealed to an attacker through this traffic.

Since the servers do not respond to messages, and incoming message do not actually contain real content, the Servers module simply receives messages from the Relayer module and logs their characteristics. The Servers module could also potentially receive messages from the Defender (or Dummy Traffic Generator) module and the Attacker module. If a message represents legitimate CONTRA traffic, the Server module performs a lookup on the intended Server destination and port number to make sure they match. It then records all information about the packet to the log file.

Rather than provide statistics as the simulation runs, I analyze the log file when the simulation ends. From the log file, I can determine the number of successful messages the Server module receives.

4.8 Simulating an Attack

The Attacker module represents an outside attacker. The simulation models the dynamics of an attack on the system followed by the system's response. I assume an attacker has the capability to launch an attack and the CONTRA system has the capability to recognize an attack. An attack would consist of a directed flood of packets to a particular address or set of addresses. A response from CONTRA would involve recognizing a load imbalance and switching the address bindings across the overloaded ports. Rather than actually create a flood of messages and recognize a load imbalance, I allow the Attacker module to simply send a message to the Servers
module indicating that an attack against a set of addresses has transpired. The Server module responds by recording the attack and changing server ports. Currently the Attacker module has no means of conducting traffic analysis. I conduct all traffic analysis in Matlab after the simulation has run. However, the design of the simulation is such that if a user were to create an attacker module, with an analysis engine, the module could be easily inserted into the system.

4.9 Summary of Simulation Parameters
Table 4.1: A list and description of all the available simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Servers</td>
<td>The total number of servers within the simulated CONTRA facility</td>
</tr>
<tr>
<td>Number of Clients</td>
<td>The total number of clients from which outbound traffic is simulated.</td>
</tr>
<tr>
<td>Addresses Per Server</td>
<td>The number of addresses each server is assigned.</td>
</tr>
<tr>
<td>Total Address Space</td>
<td>The total number of addresses available for assignment.</td>
</tr>
<tr>
<td>Total Messages</td>
<td>The simulation is run until this number of client messages are generated.</td>
</tr>
<tr>
<td>Length Type</td>
<td>A message can be a “long message” or a “short message”</td>
</tr>
<tr>
<td>Short Message Length</td>
<td>The length of a short message.</td>
</tr>
<tr>
<td>Long Message Length</td>
<td>The length of a long message.</td>
</tr>
<tr>
<td>Lower Bound Time Distribution</td>
<td>The lower bound on the Poisson parameter for traffic distribution to a server.</td>
</tr>
<tr>
<td>Upper Bound Time Distribution</td>
<td>The upper bound on the Poisson parameter for traffic distribution to a server.</td>
</tr>
<tr>
<td>Rate of Distribution Change</td>
<td>The rate of traffic distribution change</td>
</tr>
<tr>
<td>Change Addresses</td>
<td>A boolean indicating whether or not servers should change addresses throughout the course of the simulation.</td>
</tr>
<tr>
<td>N</td>
<td>The value N for a message.</td>
</tr>
<tr>
<td>K</td>
<td>The value K for a message.</td>
</tr>
<tr>
<td>M</td>
<td>The number of addresses assigned to each server.</td>
</tr>
<tr>
<td>Random Padding</td>
<td>The size of the random padding added to a block of a message</td>
</tr>
<tr>
<td>Contra Header Length</td>
<td>The length of the the CONTRA header</td>
</tr>
<tr>
<td>Relay Delay</td>
<td>The distribution of relay delays to be added to the arrival time of a message block.</td>
</tr>
</tbody>
</table>
Chapter 5

Methods and Results

The previous chapters described the parameters of both the Attacker Problem and the simulation used to analyze the Attacker Problem. The purpose of this study is to understand how the CONTRA system may manipulate its parameters to prevent an attacker from correlating addresses to servers. This chapter describes how I use the simulation to generate wide range of possible CONTRA operating conditions and configurations. I focus on a particular attack strategy and analyze an attacker’s likelihood of success. Based on this analysis, I show how various system parameters may be manipulated to impede an attacker.

5.1 Data Collection

As described in Section 3.4.2, the attacker observes network traffic entering a host facility and may use the characteristics of the traffic to perform analysis. Thus to collect data on behalf of a hypothetical attacker, I used the CONTRA simulation to generate network traffic. By manipulating the parameters of the simulation, I could produce multiple traffic histories under a wide range of CONTRA configurations.

After the CONTRA simulation is run, it produces an output log of the network traffic created during the run. This log contains all the characteristics of the traffic that would be observable to an attacker as well as information that would be encrypted and privy only to actual CONTRA hosts.
Ideally, the attacker’s analysis would be simulated as the actual CONTRA simulation is run. I, however, perform the analysis after the simulation is run by parsing the log file with Matlab and calculating attack statistics. The advantage of separating the two components is that it allows the statistics to be quickly and easily collected over a broad range of parameter values.

5.2 Attacker Strategy: Producing Correlations

As the attacker observes incoming network traffic, his aim is to somehow correlate addresses and identify groups that are likely to belong to the same server. After a preliminary study of the CONTRA system, it was revealed that a strategy based on the co-occurrence of network addresses was useful in achieving the attacker’s objective. I thus focus my analysis on this strategy and study how effective it is under a wide range of both attacker and CONTRA parameters.

The attacker strategy I simulate is as follows:

1. The attacker chooses a random address, which I call the pivot address, and finds all the occurrences of the address in the traffic history.

2. Around each occurrence of the address, he creates an interval spanning $\pm t/2$ before and after the occurrence.

3. He compares a set of $k$ consecutive intervals to generate a set of addresses that are common to all $k$ intervals.

Remember that each CONTRA message is sent in pieces, where each piece is sent to a different address. The rationale behind this strategy is that each occurrence of the pivot address represents a piece of some message being sent to a CONTRA host. Around the occurrence of that address, the attacker is likely to find other addresses to which pieces of the same message are being sent. He will also find addresses to which completely unrelated messages are being sent. By taking the common addresses among a set of $k$ intervals, he can increase his likelihood of finding a group of addresses that belong to the same server.
5.3 Attacker Probabilities

The output of the attacker's analysis is a set of addresses that may or may not belong to the same server. From this set of addresses, the attacker will want to target a certain number, \( C \), of them in hopes of hitting a group of size \( G \) that belong to the same server. It is not known how many addresses, \( C \), the attacker is capable of hitting. The larger \( C \) is, the more resources the attacker will have to devote to his attack, the more diluted the attack becomes, and the greater his risk detection. The size, \( G \), of the group the attacker will want to find will also vary with the attacker's capabilities and objectives. From a CONTRA viewpoint, the attacker will need to hit a group of size greater than \( N - K \) to result in message loss. Upon understanding the efficacy of the attacker's strategy, CONTRA may manipulate \( N \) and \( K \) to avoid message loss.

To create a set of attacker probabilities, I vary both \( G \) and \( C \) and perform the analysis described in the previous section. There will be multiple sets of \( k \) consecutive addresses the attacker could use to perform his analysis. For each set, I calculate the probability of the attacker finding a group of size \( G \) if he floods \( C \) addresses, where \( C \geq G \). The final probability I report is produced from an average over all possible sets of \( k \) intervals.

Appendix B describes the algorithm I derived to calculate these probabilities and the combinatorics behind it. Additionally, it provides a description of the Matlab tools I created to run the analysis.

5.4 Results

I performed the data collection and analysis described above by running the simulation under a set of default parameters, while varying a particular parameter of interest. I begin by presenting the default set of parameter values. I then show how the attacker's probability of finding a group of size \( G \) addresses varies with various parameters of the attacker strategy and the CONTRA system. For a review of the
simulation parameters, refer to section 4.1.

5.4.1 Default Parameters

Table 5.1 describes the default parameters used in the following experiments. This table does not contain all the parameters listed in Table 4.1. While the simulation allows for analysis on all the listed parameters, I focus on a particular subset of them. The following parameters have been left out:

- Number of Clients: The model for producing traffic does not produce traffic based on the number of clients in the system. Rather it models the aggregate traffic from all clients to a particular server. A new model for creating client traffic that uses this parameter could be substituted into the existing simulation. The tools created by this project could be used to study new models of traffic generation and their parameters.

- Message and Packet Length Characteristics: I leave out all parameters relating to length. I assume that CONTRA can easily whiten any variance that may appear due to differences in packet lengths.

- K: K does not manifest itself through the network traffic profile, therefore it is not necessary to specify a value of K. K however is useful when running the simulation as a dynamic game between CONTRA and the attacker. K determines how successful an attacker will be after launching an attack.

- Rate of Distribution Change and Address Change: I set the rate of change on behalf of the CONTRA servers to zero. I consider these parameters to be “dependent” parameters. CONTRA can set these parameters after understanding the attacker’s strategy.

5.4.2 Attacker’s Analytical Time Frame

The attacker has three parameters available to him to adjust when performing his analysis on CONTRA network traffic:
1. $t$: The attacker performs his analysis by finding all occurrences of a random address and creating time intervals $\pm t/2$ around the occurrences. His choice in $t$ will affect the noise produced by his analysis.

2. $k$: Once the attacker generates time intervals, he chooses $k$ consecutive subsets of time intervals to compare. Like his choice of $t$, $k$ will affect his ability to successfully find addresses from a single server.

3. $C$: After the attacker has found the common addresses among a set of $k$ intervals, he must choose $C$ of those addresses to flood. The more addresses he targets, the more likely he is to hit addresses from the same server. However, he must also devote more of his resources to the attack to avoid diluting it, thus increasing his chance of detection.

Here I explore how the attacker can improve his strategy by manipulating $C$, $t$, and $k$. I run the simulation using the default parameters and graph how the attacker’s probability of success changes with varying values of $C$, $t$, and $k$.

**Variations in $C$**

Figure 5-1 plots the attacker’s probability of finding some group of size $G$ for different values of $C$. In this experiment, the largest size group the attacker would be able to hit with some nonzero probability is a group of size 3. His probability of hitting a
group of 3 increases dramatically if he increases his choice of $C$ from 5 to 6. If he
directs his attack at more than 6 addresses, he guarantees that he finds a group of
size 3.

In general, the obvious trend is that as he increases $C$, he increases his probability
of finding any group of size $G$. This makes sense and can be expected to generally be
true regardless of how the other parameters of the system behave.

It is important to note that in a special circumstance, an increase in $C$ may
actually cause the attacker's probabilities to fall to zero. If the attacker performs
his analysis and does not produce a set of at least $C$ addresses to choose from, his
probability of finding any group of $G$ will be zero. Therefore, his probabilities for
small values of $C$ may be nonzero, while his probabilities for large values of $C$ are
zero. I call this the pruning effect.

![Graph showing variations in $C$ for $t = 2$ and $k = 2$.]

Figure 5-1: Variations in $C$, for $t = 2$ and $k = 2$

**Variations in $t$**

In the previous section, I set the value of $t$ to 2 and $k$ to 2. In this section I keep
$k$ constant, and vary $t$ for two values of $C$. Figures 5-2a and 5-2b depict how the
attacker's probabilities change with variations in $t$ for $C = 10$ and $C = 5$, respectively.
As the attacker increases $t$ from zero, he increases his probabilities significantly. But very quickly, after reaching some optimal point, his probabilities begin again to decrease. Figure 5-2c illustrates this for $G = 3, C = 5$ and $G = 3, C = 10$. It also suggests that for $k = 2$, his probabilities are optimized when $t = 3$.

![Figure 5-2c: Attacker Probabilities with Variations in $t$.](image)

Variations in $k$

Now I explore how varying $k$ affects the attacker's strategy. In the previous experiments, $k$ was fixed to 2. Since the previous experiment suggested that $t = 3$ is an optimal strategy, I vary $k$ while holding $t$ constant at 3. Figure 5-3a shows that by increasing $k$ beyond 2 the attacker’s probabilities immediately fall to zero. This is the result of the pruning effect (see 5.4.2). As the attacker increases $k$, he fails at finding
any common addresses beyond his pivot address. Thus his probabilities of finding a

group $G > 1$ fall to zero.

Figure 5-3b suggests that for each value $k$, there is some optimal $t$. $k$ is fixed to

5, as $t$ varies. Consistent with the previous figure, for low values of $t$, the attacker

probabilities are zero. However, if $t$ is increased to beyond 10, the probabilities become

nonzero and comparable with probabilities produced with $k = 2$ and $t = 3$.

Figure 5-3c fixes $C = 10$ and $G = 5$ and graphs the probabilities achieved by

varying both $k$ and $t$. For large values of $k$ and small values of $t$ the probabilities

are zero. But for large values of $k$ and large values of $t$, the probabilities increase

significantly.
Figure 5-3: Attacker Probabilities with Variations in k.
Optimal Attacker Strategies

From this analysis, the attacker appears to have two dimensions to his strategy. He can choose to perform his analysis on a short time scale or a long time scale. With a short time scale strategy, he chooses a small value of $t$ and a small value of $k$. With a long time scale strategy, he chooses relatively large values of both $t$ and $k$.

The attacker may also choose $C$, the total number of addresses he wishes to target. As demonstrated above, in general the more addresses he chooses the more he increases his likelihood of finding some subset of size $G$. However, because of the pruning effect, his analysis may produce a total set of addresses of size less than $C$. In that case, his probabilities will fall to zero.

The attacker’s choice of strategy will be a reflection of his resources. If the attacker chooses the long time scale strategy, he will have to devote more time to watching the traffic and analyzing it before launching an attack. If he increases his value of $C$, he must devote more of network resources to flooding CONTRA at the risk of being detected.

As I continue to investigate the parameters of the CONTRA system, it is useful to explore both dimensions of the attacker’s strategy. I thus perform my subsequent analysis for $C = 5$ and $C = 10$, both using a long and short scale time strategy. For a short time scale strategy, I fix $k = 2$ and $t = 3$. For a long time scale strategy, I fix $k = 8$ and $t = 20$. As I change the other parameters of the simulation, one strategy may prove more effective than another. It may also be that additional use of the attacker’s resources will prove unnecessary for the attacker to successfully achieve his objectives.

5.4.3 Traffic Distributions

The simulation uses a model for traffic generation that has two main parameters. The first parameter is the rate at which traffic is generated to a particular CONTRA server. Each server is assigned some exponential parameter that describes the rate of message generation to that server. After the simulation generates a messages to a
server, the message is broken up and each piece is sent to a server with a relay delay. The delay is chosen from a distribution that is also characterized by an exponential parameters.

This section investigates how the attacker’s probabilities vary with these parameters. For a full review of the traffic generation model, refer to Section 4.5.

Distribution Bounds

Apart from manipulating the relay delay, CONTRA does not have any direct control on the time distributions of traffic to its servers. Still it is useful to understand how the attacker’s strategy is affected as the possible spread of the rate of traffic to it’s servers becomes more disparate. In Figure 5-4, I vary the possible lower and upper bounds of the traffic rates and plot how the attacker’s probabilities changes. In 5-4a, 5-4b, 5-4c, and 5-4d the distributions are described by the lower and upper bound pair, $(l, u)$. In 5-4e and 5-4f, $G$ and $C$ are fixed, and the probabilities are plotted against the differences in the lower and upper bounds.

As the difference between $l$ and $u$ increases, the traffic becomes less homogenous. On the long time scale, Figures 5-4b and 5-4d demonstrate that the less homogenous the traffic is, the easier it is to correlate addresses. This makes intuitive sense. This trend is less pronounced on the short time scale. In all four cases, the attacker performs best when $(l, u) = 15$.

Additionally, when $C = 10$, for high values of differences, the attacker’s probabilities fall to zero. This is reflection of the pruning effect. The attacker is unable to find a set of 10 common addresses to choose from. However it indicates that as the traffic becomes less homogenous, the attacker is able to devote less of his resources to the attack while achieving fairly good probabilities.

Relay Delays

Figure 5-5 illustrates how the attacker’s probabilities change with varying relay delays. In each scenario, as the exponential parameter for the relay delay becomes longer, the attacker is less successful in correlating addresses. In general, the attacker is most
successful when the spread of the relay delays is as small as possible. This makes sense because the attacker would be able to see the more pieces of particular message around his pivot address in his chosen time interval.

As the figures demonstrate, when the relay delay is small, the long time scale strategy is much more effective than the short time scale strategy. However as the delay increases, the attacker does not achieve significant gains by devoting more time to his analysis.
Figure 5-4: Attacker Probabilities with Variations in Traffic Distributions.
Figure 5-5: Attacker Probabilities with Variations in Relay Delays.
5.4.4 Addresses to Address Space Ratio

Each CONTRA server distributes its identity among a set of \( m \) IP addresses in a large IP address space. The ratio of the number of addresses in use to the total number of addresses available varies with, \( S \), the total number of servers, and \( m \) the total number of addresses per server. In this section, I explore how the values of these parameters affect attacker’s probabilities.

Addresses Per Server

Figure 5-6 graphs how the attacker’s probabilities vary with the value of \( m \), the number of addresses per server. With a long time scale strategy, Figures 5-6b and 5-6d illustrate how the attacker’s strategy becomes susceptible to the pruning effect. As \( m \) increases, the attacker is unable to find any common addresses. However, increasing \( m \) is not necessarily to the attacker’s disadvantage. Figures 5-6a and 5-6c suggest that increasing \( m \) may actually assist the attacker. In 5-6c for \( m = 30 \) and \( m = 35 \), before the pruning effect drops the attacker’s probabilities to zero, the attacker’s probabilities actually increase with increasing \( m \). His probabilities of finding a group of 3 and 4 become 1.

By increasing \( m \), a CONTRA host has more addresses to choose \( N \) from when sending a message. Figure 5-6 suggests that by increasing \( m \), actually aids the attacker. It becomes harder to find addresses that are common among the attacker’s choice of \( k \) intervals. But when attacker does find a set of common addresses, it is more likely that they belong to the same server. Thus the attacker has to devote less of his resources to analyzing the traffic and launching an attack.

Total Number of Servers

Figure 5-7 graphs how the attacker’s probabilities change with increasing the number of servers in the host facility. There is a clear trend that as the total number of servers increases, the attacker’s probabilities fall quickly and significantly. This makes intuitive sense. As the number of servers increases, the attacker is likely to see more
network traffic into the host facility during his period of analysis. The more traffic he sees, the more common addresses he finds in his chosen set of $k$ intervals. Thus it becomes harder for him to correlate addresses.

It is interesting to note, that as the number of servers increase, both the long and short time scale strategies perform similarly. Also, as the number of servers increase, the attacker's gains from devoting more of his own network resources to the attack become less significant.
Figure 5-6: Attacker Probabilities with Variations in Addresses Per Server.

(a) C = 5, Short Time Scale Strategy

(b) C = 5, Long Time Scale Strategy

(c) C = 10, Short Time Scale Strategy

(d) C = 10, Long Time Scale Strategy

(e) G = 3, C = 10, Comparison of Strategies with Varying Addresses Per Server

(f) G = 5, C = 10, Comparison of Strategies with Varying Addresses Per Server
Figure 5-7: Attacker Probabilities with Variations in Number of Servers.
5.4.5 K-of-N Encoding

In order to be sent to another CONTRA host, a CONTRA message must be encoded with redundancy and broken up into \( N \) pieces. Each piece is sent to a unique address chosen from the set of \( m \) total addresses belonging to the destination host. Figure 5-8 graphs how the attacker’s probabilities change with varying values of \( N \).

For low values of \( N \), the attacker’s probabilities fall to zero. Of course, for low values of \( N \) the attacker does not need to find very large sets of addresses to target. If \( N \) is low, the attacker can disrupt a server by simply targeting one or two addresses. Thus, \( N \) should not be kept low.

After initially increasing, the attacker’s probabilities begin to decrease slightly as \( N \) approaches \( m \). Keeping \( N \) high is beneficial to CONTRA because messages can be sent with added redundancy and thus the system becomes resistant to attacks. CONTRA should not however, set \( N \) to equal \( m \). If \( N \) equaled \( m \), then the attacker could simply count occurrences of addresses. Those with the same number of occurrences could be easily identified as belonging to the same server.
Figure 5-8: Attacker Probabilities with Variations in N.
5.4.6 Dummy Traffic

Figure 5-9 depicts how the attacker's probabilities change with the introduction of dummy traffic. Dummy traffic is introduced by assuming that some percentage of the addresses produced by the attacker's analysis belong to dummy servers. Under all of the attacker strategies depicted in Figure 5-9, the attacker's probabilities fall rapidly as dummy traffic is produced. With just a 10% level of dummy traffic, CONTRA is able to significantly impede the attacker without overusing its network resources. An increase in network resources to generate dummy traffic beyond a level of 20% becomes insignificant in affecting the attacker. Thus CONTRA can devote a minimal level of resources to producing dummy traffic and successfully impede an attacker.
Figure 5-9: Attacker Probabilities with Variations in Dummy Traffic Percentages.
Chapter 6

Conclusion

In this Masters Thesis project, I have made the following contributions:

1. I have provided a definition of the Attacker Problem as related to the CONTRA system.

2. I have designed and created a modular simulation for studying the CONTRA system. This simulation is parameterized and can be run under a wide range of conditions and configurations. The main components of the simulation can also be substituted with new modules, if new models for modeling the CONTRA system become available.

3. I have defined a statistic to measure the success of an attacker and provided an algorithm to calculate the statistic. I have also created a tool that implements the algorithm.

4. I have studied the CONTRA system under a particular attacker strategy and calculated statistics under a wide range of varying parameters. From my study, I have produced the following recommendations:

(a) Traffic to individual CONTRA hosts should be whitened as much as possible to make the rate of traffic to all hosts homogenous. The more disparate the rates are, the easier it is for an attacker to correlate addresses.
(b) Keep the spread of relay delays long. The longer they are, the more resources an attacker must devote to analyzing the traffic.

(c) Encourage increases in the number of total servers. This makes it harder for the attacker to find addresses belonging to a particular server. If a host facility has very few CONTRA servers, it will be relatively easy for the attacker to target them.

(d) Keep the $N$ to $m$ ratio above 50%. As $m$ increases, leaving $N$ constant, the attacker can achieve his objectives while devoting less resources to the attack.

(e) Devote network resources to generating dummy traffic. A level of dummy traffic between 10% and 20% of the total network traffic entering the facility is enough to significantly impede the attacker. There are no significant gains from generating dummy traffic above this level.

The analysis demonstrates that the CONTRA system is fairly robust to outside attacker analysis.
Appendix A

CONTRA Simulation User Manual

The creation of an extensible simulation of the CONTRA system was central to this study. The following serves as a user manual for this tool so that it may be used to conduct further analysis of the CONTRA system. The tool is modular and each module has configurable parameters. This study made many assumptions about the nature of Internet network traffic and the behaviors of the CONTRA system. As new models become available, they may be substituted into the simulation to analyze the resultant behavior of the system as a whole.

A.1 Overview

The simulation is implemented in OMNeT++, an object-oriented, discrete event simulator. This manual provides a summary of the components of the simulation so that it may be easily run or modified. While this overview will be useful in understanding the simulation, it may be necessary to refer to more detailed information about OMNeT++ simulations, available at http://www.omnetpp.org/.

The simulation is organized into modules, depicted in Figure 4-1. Each module is defined by a name, a list of parameters, and a list of input and output gates. Gates are used to connect modules together to create inter-modular communication channels. Messages are sent from one module to another along these channels. In this simulation, messages usually represent network traffic.
The CONTRA simulation has five main components. Before running or modifying the simulation, it is essential to understand them. They are:

1. the topology files specified by a "ned" extension
2. the message definitions
3. the module source files, written in C++
4. the configuration file, omnetpp.ini
5. the log files, specified by a "vec" extension

The following sections describe each of these components in more detail.

A.2 Topology Files

Any module that is used in the simulation must be defined with a name, a list of parameters, and a list of input and output gates. The "ContraModules.ned" file contains all the module definitions for the simulation. The format of the file is straightforward and it is easy to modify existing modules or create new ones by duplicating the structure. C++ files, named according to the module they represent, contain the actual source code for each module and are discussed later.

Once a module is defined, it must be connected to other modules. The "Contra.ned" is the topology file that describes how modules are linked together. The steps to creating and linking modules are as follows:

1. Create and name instances of each of the defined modules.
2. Assign values to the parameters in each module.
3. Connect input and output gates.

To substitute new modules into the simulation, existing modules can be "disconnected" from the other modules, and new modules can be put in place in a similar manner.
A.3 Message Definitions and Communication Channels

Inter-module connections create directed communication channels between modules. Modules relay messages along these communication channels. When run, the simulation initializes time to be zero and from thereon, modules receive messages at discrete points in local simulation time.

In the CONTRA simulation, messages originate from the “Traffic Generator” module. The traffic generator schedules communication between hypothetical CONTRA “clients” and CONTRA “servers.” OMNeT++ allows messages to be sent as an object from a predefined message class, cMessage, or a subclass of cMessage with custom fields. The message sent by the Traffic Generator is a subclass of cMessage, named GeneratorMsg with fields specified in the file “ContraMsg.msg.”

Every time the Traffic Generator schedules communication to a server, it sends the message to the Relayer module. The job of the Relayer is to send communication to the servers according to the CONTRA protocol. Every time the Relayer receives a message from the Traffic Generator, it does a lookup on the Infoserver for the addresses of the specified server. It sends multiple messages to the Servers module, each of the type PacketMsg also defined in the ContraMsg.msg file. Each message represents a single block of the $N$ total blocks comprising the message.

Additionally, communication channels are present from the Defender to the Contra Relayer module and from the Attacker to the Servers module. These communication channels could be used to generate dummy traffic from the Defender module and to allow for attacks on the servers.

It is important to note that no message based communication is conducted with the Infoserver. The Infoserver is essentially treated as a data structure that every module has a copy of in order to request information or changes in server/address bindings. Conceptually, any module communicating with the Infoserver should carry out communication through legitimate OMNeT++ communication channels. For efficiency reasons, however, a pointer to the Infoserver is given to each module.
A.4 Modules

C++ source files contain the actual code for each module. Their names correspond to the names of the modules they represent. Each module class subclasses the cSimpleModule class provided by OMNeT++. In order for the subclass to be useful, it must redefine one or more of the following virtual member functions of cSimpleModule:

- void initialize()
- void handleMessage(cMessage *msg)
- void activity()
- void finish()

When the simulation begins, it calls the initialize function of each module. The initialization function should include any activity a module must complete before the start of the simulation. The current modules accomplish a subset of three main tasks in the initialization period, depending on their needs. First, they wait until the Infoserver completely initializes and then initialize their own pointer to the Infoserver. Then they read in and store values for the parameters specified in the module definitions. Then finally they initialize any important data structures and open log files for recording data during the simulation. The Traffic Generator additionally uses the initialization to begin scheduling the first messages to be sent to the Relayer.

A module’s handleMessage or activity functions process incoming messages (events). A module may implement either handleMessage or activity, but not both. The functions represent different event handling models. A module invokes handleMessage only when it receives an incoming messages, whereas activity is run as a coroutine. It is preferable to use handleMessage because it consumes less memory and scales better than the activity function. As such, all existing modules employ handleMessage.
The finish method deconstructs the module and frees any necessary memory. Beyond these functions, any other helper functions may be created within the module's class. The Infoserver, in particular, implements several other functions.

A.5 Event Logging

OMNeT++ allows time series data to be logged to an output text file. Any information about a received packet or other event, can be recorded to data objects, called cOutVectors, provided by OMNeT++. The cOutVector's record() method takes in a piece of data and writes it along with a time stamp to file. A single simulation may use multiple cOutVectors to record data. All data, however, is written to a single output file, "omnetpp.vec." This name may be changed in the configuration file.

Each cOutVector is characterized by a name and a description. A single cOutVector records data of a single type. In the CONTRA simulation, the Server module is the only module that logs events. Every time a packet is received, the module records each field of the packet to a cOutVector.

A typical output file may look like the following:

vector 0 "contra.servers" "client" 1
0 1.80228399 1
vector 1 "contra.servers" "server" 1
1 1.80228399 1
vector 2 "contra.servers" "address" 1
2 1.80228399 7
vector 3 "contra.servers" "serverLookup" 1
3 1.80228399 1
vector 4 "contra.servers" "n" 1
4 1.80228399 10
vector 5 "contra.servers" "k" 1
5 1.80228399 7
The output file contains either vector lines or data line. A vector line, identified by the keyword "vector," introduces a vector. It provides the number of the vector (a numerical identifier), the name of the vector, and the vector's description. A data line describes a single piece of data by the vector number, the time stamp, and the data value.

In order to parse the output and read in information for a single event, all vectors with identical time stamps must be grouped together. I created a Matlab tool to read and parse the file in Matlab. The tool's name is ReadOmnnetData and may be run from the Matlab prompt with the following arguments:

- the name of the vector file without the .vec extension
- a boolean indicating whether to output a Matlab data file
- the name of the Matlab data file to output

If the boolean argument is set to true, the tool outputs the data to a Matlab data file with the output name concatenated with the current day and time.

A.6 Configuration and Parameters

The name of the simulation's configuration file is "omnetpp.ini" by default. The configuration file has the following clearly labeled sections:
• [General] This section contains settings that apply to the entire simulation, the most notable of which are the maximum simulation and cpu usage time allowed. Other settings may be specified to optimize the performance of the simulation.

• [Cmdenv] This section contains settings that apply only when the simulation is run from the command line.

• [Tkenv] This section contains settings that apply only when the simulation is run from the graphic user interface.

• [Parameters] This section specifies the values that should be assigned to the various module parameters. The ability to specify values in the configuration file is useful because different values may be substituted into the simulation without having to recompile the source code. The parameters may be assigned numerical, Boolean, or random values. If a random distribution is assigned to a parameter, then the parameter's value changes every time it is read in. See section 4.1 for a list of available parameters and their descriptions.

• [Runs] When the simulation is run from the command line, multiple runs may be executed in batch, each with different parameter values. This section allows the user to define multiple runs and assign different values for each run.

For additional information, refer to the OMNeT++ manual.

A.7 Compiling and Running

To compile the simulation to run from the graphical user interface, type the following into the command line:

    opp_makemake -u Tkenv

or, to compile the simulation to run from the command line, type:

    opp_makemake -u Cmdenv
The simulation is run from the command line by typing \texttt{./contra}. If the simulation is run with the user interface, the OMNeT++ graphical interface will load with a visualization of the modules in the simulation and how they are connected. The simulation may be run in either express or normal mode, where express mode omits the network visualizations and normal mode displays them. The express mode is most efficient and practical for running the simulation for any significant course of time.
Appendix B

Explanation of Combinatorics

B.1 Problem Statement

The input to the problem is a set $S$ of addresses. Each address maps to a server. Within $S$ we have subsets, $S_{1,2},...,S_n,$ of addresses that belong to the same server. For example, the input may be the following

$$S = \{\{15, 999, 3, 87\}, \{872, 66, 231\}, \{721, 292, 33, 2, 70, 4\}, \{67\}\}$$

Alternatively this input could be described as $S'$ by the number of elements in each subset:

$$S' = \{4, 3, 6, 1\}$$

Each element in the above description describes the number of addresses that belong to the same the original input.

If the attacker chooses $C$ addresses from the entire input, we want to know his chance of finding a subset of at least $G$ addresses, where $G \leq C$, belonging to the same server. Thus we want to calculate $P(G|C)$.

This probability can be computed by counting the total number of sets of size $C$ which contain subsets of at least size $G$, and dividing by the total number of sets of size $C$. The total number of ways to choose $C$ addresses from the input, where the number of addresses in the input is $N$, is simply $\binom{N}{C}$. I now present two solutions
for counting the total number of sets of size $C$ that contain a subset of at least $G$ addresses that belong to the same server.

## B.2 Solution A

In order to calculate the total number of sets of size $C$ that contain a subset of at least size $G$, we can instead count the number of sets that contain no subsets of size $G$. By subtracting this number from the total number of ways to choose $C$ addresses, we can achieve the desired number of sets.

To count the number of sets that do not contain any subset of at least $G$, we can take at most $G - 1$ addresses from any subset of addresses that belong to the same server. The following algorithm produces the number of sets of size $C$ that do not contain any subset of $G$ or more addresses from the same server:

- Suppose the input the problem is of the form:
  
  $$S' = \{N_1, N_2, \ldots, N_k\}$$

  where each element $N_i$ represents the number of addresses belonging to a particular server.

- Initialize the total count to zero.

- Begin with the first element, $N_1$, in the input set.

- We want to augment the total count of sets to equal the number of sets produced by choosing 0 to at most $G - 1$ elements from this set.

- Perform the following: For $j = 0 \ldots \min((G - 1), N_1)$
  1. Count the number of combinations produced by choosing $j$ elements from the set. This is simply $\binom{N_1}{j}$.
  2. Multiply this number by the number of combinations produced by choosing at most $C - j$ elements from the rest of the sets. This is calculated by calling
the algorithm recursively with the rest of the input. Add the product of this multiplication to the total count.

- Sum the total number of sets produced in this fashion and return the total.

The above algorithm recursively calculates all possible sets by counting the number of combinations produced by taking anywhere between 0 to \( G - 1 \) addresses from any given subset. While this produces the correct solution, it is not an efficient way to count the number of sets. The next section provides a solution with a more acceptable runtime.

### B.3 Solution B

We can count the number of sets of size \( C \) that contain subsets of at least size \( G \) as follows:

- Suppose the input the problem is of the form:

\[
S' = \{N_1, N_2, \ldots N_k\}
\]

where each element \( N_i \) represents the number of addresses belonging to a particular server.

- If we choose any \( C \) addresses from the input, define the notion of a 'win' and a 'loss' for a subset of addresses belonging to a common server. A 'win' from a particular subset is a situation where there are \( G \) or more elements from that subset. Conversely, a 'loss' occurs when there are less than \( G \) elements from a particular subset.

- Iterate through all the elements in \( S' \). Thus, for \( i = 0 \ldots |S'| \):

  - Keep track of the following:

    1. **count**: the total count of sets of size \( C \) that contain a subset of at least size \( G \)
2. visitedSets: the set of elements in the input already visited
3. \( N_i \) the current element in the input
4. futureSets: the set of elements in the input not yet visited

- For \( N_i \), count the number of ways we can choose \( C \) elements from the total input by achieving a "win" from the \( N_i \). If \( N_i < G \), then this is simply zero. Otherwise:

  * To achieve a "win" from \( N_i \), we can choose \( G, G + 1, \ldots, \min(N_i, C) \).
  Thus for \( j = G \ldots \min(N_i, C) \):

    1. Count the number of ways to choose \( j \) elements from \( N_i \). This is \( \binom{N_i}{j} \).
    2. Multiply this number by the number of ways to choose the rest of the \( C - j \) addresses from the rest of the input in the following manner:

  - Addresses chosen from visitedSets must be chosen in such a way that no "wins" are achieved from any subset in visitedSets. Thus, at most \( G - 1 \) elements may be chosen from any set in visitedSets.
  - Addresses from futureSets may be chosen in any manner.
  - The total possible combinations of elements chosen from visitedSets and futureSets can be enumerated recursively with a trivial algorithm similar to that presented in Solution A.

  * Add the total number of sets produced to the total count.
Bibliography


