

# Algorizmi: A Configurable Virtual Testbed to Generate Datasets for Offline Evaluation of Intrusion Detection Systems

by

Karim Ali

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Master of Mathematics  
in  
Computer Science

Waterloo, Ontario, Canada, 2009

© Karim Ali 2009

## **AUTHOR'S DECLARATION**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Karim Ali

## Abstract

Intrusion detection systems (IDSes) are an important security measure that network administrators adopt to defend computer networks against malicious attacks and intrusions. The field of IDS research includes many challenges. However, one open problem remains orthogonal to the others: IDS evaluation. In other words, researchers have not yet succeeded to agree on a general systematic methodology and/or a set of metrics to fairly evaluate different IDS algorithms. This leads to another problem: the lack of an appropriate IDS evaluation dataset that satisfies the common research needs. One major contribution in this area is the DARPA dataset offered by the Massachusetts Institute of Technology Lincoln Lab (MIT/LL), which has been extensively used to evaluate a number of IDS algorithms proposed in the literature. Despite this, the DARPA dataset received a lot of criticism concerning the way it was designed, especially concerning its obsolescence and inability to incorporate new sorts of network attacks.

In this thesis, we survey previous research projects that attempted to provide a system for IDS offline evaluation. From the survey, we identify a set of design requirements for such a system based on the research community needs. We, then, propose Algorizmi as an open-source configurable virtual testbed for generating datasets for offline IDS evaluation. We provide an architectural overview of Algorizmi and its software and hardware components. Algorizmi provides its users with tools that allow them to create their own experimental testbed using the concepts of virtualization and cloud computing. Algorizmi users can configure the virtual machine instances running in their experiments, select what background traffic those instances will generate and what attacks will be launched against them. At any point in time, an Algorizmi user can generate a dataset (network traffic trace) for any of her experiments so that she can use this dataset afterwards to evaluate an IDS the same way the DARPA dataset is used.

Our analysis shows that Algorizmi satisfies more requirements than previous research projects that target the same research problem of generating datasets for IDS offline evaluation. Finally, we prove the utility of Algorizmi by building a sample network of machines, generate both background and attack traffic within that network. We then download a snapshot of the dataset for that experiment and run it against Snort IDS. Snort successfully detected the attacks we launched against the sample network. Additionally, we evaluate the performance of Algorizmi while processing some of the common usages of a typical user based on 5 metrics: *CPU time*, *CPU usage*, *memory usage*, *network traffic sent/received* and *the execution time*.

## Acknowledgements

Thanks to Allah, The One, The Most Merciful, The Most Gracious and The Sovereign of the Day of Judgment. Thanks to Allah for giving me such a great opportunity to continue my graduate studies at a reputable school. I thank Him for always being there for me, especially when I am weak, or having hard time.

I am grateful to my parents for helping me continue my graduate studies abroad. I really have no idea how I could have done it without your help. Words can never really express how much I love you and respect you. I promise I will always keep you in my prayers. May Allah bless you and reward you for your patience, perseverance and unconditional love.

To Professor Abdelaziz Ismail, if we had something similar to the notion of *The Godfather* in our culture, then you would be my godfather. You know how you pushed the idea of joining AUC into my head (and my parents' as well) which I think eventually led me (thanks to Allah) to the life I have right now. Thank you so much!

To Sarah, words can never describe how you changed my life. I can never (and would not like to) imagine my life without you. You've always been there for me during the hard times when I wanted to feel that I'm not facing all those difficulties alone. You were always there to share our very precious moments of joy and happiness, and they are a lot elhamdulillah :) with more to come inshallah ;) .. I love you.

I would like to thank my dearest friends Afify, Se7s, Army, Andy, Hamzaway, Meezo and Ismail for being there for me all the time. You guys with all the other CSaweya define what true friendship is. Yes sometimes we go through hard times, but who doesn't :) I can't forget to thank my undergraduate thesis group, the beepers, I really long for those good old days.

I would like to express my appreciation to my supervisor, Professor Raouf Boutaba for his help and support throughout my Master's degree. I would also like to thank my readers, Professor Urs Hengartner and Professor Ian McKillop, for their insightful comments and constructive criticisms.

*To my dear parents who made me who I am today and to the one I love, my adorable fiancée, Sarah.*

# Contents

List of Tables . . . . .	x
List of Figures . . . . .	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Intrusion Detection Systems Evaluation . . . . .	1
1.2 Challenges . . . . .	2
1.3 Motivation . . . . .	3
1.4 Contributions . . . . .	3
1.5 Thesis Organization . . . . .	4
<b>2 Literature Review</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Chapter Organization . . . . .	6
2.3 Network Experiments . . . . .	6
2.3.1 Mathematical Models . . . . .	7
2.3.2 Simulation . . . . .	7
2.3.3 Emulation . . . . .	7
2.3.4 Real System . . . . .	7
2.4 Offline Evaluation of Intrusion Detection Systems . . . . .	8
2.4.1 Dataset Anonymization . . . . .	9
2.4.2 Dataset Generation . . . . .	9
2.5 Summary . . . . .	12

<b>3</b>	<b>Design Requirements</b>	<b>14</b>
3.1	Introduction . . . . .	14
3.2	Chapter Organization . . . . .	14
3.3	Availability . . . . .	15
3.4	Versatility . . . . .	15
3.5	Fidelity . . . . .	16
3.5.1	Background Traffic . . . . .	16
3.5.2	Attack Traffic . . . . .	16
3.6	Economy . . . . .	17
3.7	Efficiency . . . . .	17
3.8	Functionality . . . . .	18
3.8.1	Measurement Tools . . . . .	18
3.8.2	Datasets . . . . .	18
3.9	Containment and Security . . . . .	19
3.10	Usability . . . . .	19
3.11	Reproducibility . . . . .	20
3.12	Maintenance . . . . .	20
3.13	Summary . . . . .	21
<b>4</b>	<b>Algorizmi</b>	<b>23</b>
4.1	Introduction . . . . .	23
4.2	Chapter Organization . . . . .	23
4.3	Architectural Overview . . . . .	23
4.3.1	Infrastructure Layer . . . . .	24
4.3.2	Algorizmi Control Layer . . . . .	24
4.3.3	Experiment Control Layer . . . . .	25
4.3.4	Experiment Layer . . . . .	26
4.3.5	Tools Layer . . . . .	26
4.3.6	Graphical User Interface . . . . .	26

4.4	System Components . . . . .	27
4.4.1	Infrastructure Components . . . . .	28
4.4.2	Eucalyptus . . . . .	30
4.4.3	Algorizmi Manager . . . . .	31
4.4.4	Application Components . . . . .	32
4.5	Using Algorizmi to Generate a Dataset . . . . .	45
4.5.1	Create New User Account . . . . .	45
4.5.2	Modify User Settings . . . . .	46
4.5.3	Create New Experiment . . . . .	47
4.5.4	Add a VMI to an Experiment . . . . .	48
4.5.5	Add a Background Script to an Experiment . . . . .	49
4.5.6	Generate Attack Traffic . . . . .	50
4.5.7	Download Dataset . . . . .	52
4.6	Summary . . . . .	52
<b>5</b>	<b>Evaluation</b>	<b>54</b>
5.1	Introduction . . . . .	54
5.2	Chapter Organization . . . . .	54
5.3	Dataset Validation . . . . .	54
5.4	Design Requirements Satisfaction . . . . .	60
5.5	Performance Assessment . . . . .	61
5.5.1	Environment Settings . . . . .	61
5.5.2	Performance Metrics . . . . .	61
5.5.3	Results . . . . .	62
5.6	Summary . . . . .	62
<b>6</b>	<b>Conclusion</b>	<b>65</b>
6.1	Summary of Contributions . . . . .	65
6.1.1	Survey of IDS Evaluation Systems . . . . .	65
6.1.2	Algorizmi . . . . .	65



6.2	Limitations . . . . .	66
6.2.1	Using EPC . . . . .	66
6.2.2	Metasploit Integration . . . . .	67
6.3	Future Work . . . . .	67
6.4	Concluding Note . . . . .	68
	<b>APPENDICES</b>	<b>69</b>
A	<b>Algorizmi Implementation Experience</b>	<b>70</b>
	<b>References</b>	<b>86</b>

# List of Tables

2.1	Summary of related projects . . . . .	13
3.1	Compliance of previous research efforts to the Availability design requirements	15
3.2	Compliance of previous research efforts to the Versatility design requirements	15
3.3	Compliance of previous research efforts to the Background Traffic Fidelity design requirements . . . . .	16
3.4	Compliance of previous research efforts to the Attack Traffic Fidelity design requirements . . . . .	17
3.5	Compliance of previous research efforts to the Economy and Efficiency design requirements . . . . .	18
3.6	Compliance of previous research efforts to the Measurement Tools design requirements . . . . .	18
3.7	Compliance of previous research efforts to the Dataset design requirements	19
3.8	Compliance of previous research efforts to the Containment and Security design requirements . . . . .	19
3.9	Compliance of previous research efforts to the Usability design requirements	20
3.10	Compliance of previous research efforts to the Reproducibility and Maintenance design requirements . . . . .	20
3.11	Compliance of previous research efforts in the field of IDS evaluation to our identified design requirements . . . . .	22
5.1	Compliance of Algorizmi to the typical research requirements . . . . .	60
5.2	Algorizmi Performance Assessment . . . . .	64

# List of Figures

2.1	Four approaches of network experiments [93]	6
2.2	Examples of experimental tools corresponding to the four approaches of network experiments [93]	8
4.1	Algorizmi Abstract Architecture	24
4.2	Components of Algorizmi	27
4.3	A Typical Eucalyptus Installation	30
4.4	A snapshot of Algorizmi Dashboard Tab	32
4.5	A snapshot of Algorizmi Key Pairs Management Tab	34
4.6	A snapshot of Algorizmi Virtual Machine Images Management Tab	35
4.7	A snapshot of Algorizmi Background Script Templates Management Tab	37
4.8	A snapshot of Metasploit GUI	38
4.9	A snapshot of Algorizmi Experiments Management Tab	39
4.10	Authorizing a new network ingress rule	40
4.11	Adding a new background script to an experiment	41
4.12	Adding a VMI to an experiment	42
4.13	Downloading the dataset of an experiment	44
4.14	Create a new user account	45
4.15	Modify user settings	46
4.16	Create a new experiment	47
4.17	Add a VMI to an existing experiment	48
4.18	Add a Background Script to an existing experiment	49
4.19	Generate attack traffic from Metasploit	50

4.20	Metasploit attack summary . . . . .	51
4.21	Generate and download the dataset for an Algorizmi experiment . . . . .	52
5.1	Sample Algorizmi Experiment . . . . .	55
5.2	Snort alerts for the Xmas scan . . . . .	57
5.3	Snort alerts for Teardrop, Trin00 and UPD port scan attacks . . . . .	58
5.4	Snort alerts generated by the traffic of our Nessus scan . . . . .	59
5.5	Response time to user request for instantiating a set of VMIs . . . . .	63

# Chapter 1

## Introduction

Today's computer networks are subject to more attacks and intrusions than before. This is seen through analyzing the side-effects of network attacks and the losses they cause to companies. For example, [1] reports that network attacks caused some companies in the United States to lose US \$130 million in 2005. Moreover, Symantec notes in its Internet Security Threat Report for 2008 [120] that new malicious code threats reported a 136% increase in the second half of 2007 over the first half of the same year. In October 2008, Georgia Tech Information Security Center (GTISC) hosted its annual summit to discuss emerging security threats. At the conclusion of the event, GTISC released a white paper [28] that ranks malware as one of the top five security threats. The same white paper mentions that malicious programs have increased by more than 8000 programs over the period of one month. In April 2009, Symantec reports that the average cost per incident of a data breach in the United States in 2008 was US \$6.7 million, marking a 5% increase compared to 2007 [54]. Such an evolving trend of threats cannot be countermeasured using firewalls alone [1]. This fact led to investing more research efforts in the field of intrusion detection systems (IDS) to provide adequate protection for computing/networking environments.

### 1.1 Intrusion Detection Systems Evaluation

Intrusion Detection Systems (IDSes) are very important in safeguarding computer systems as the main goal of an IDS is to detect malicious/unauthorized use of resources. IDSes can be categorized according to their nature of work into two categories: signature-based (e.g. Snort [106]) and anomaly-based (e.g. Bro [97]). The former has obvious limitations in detecting novel intrusions that were not previously probed by the IDS. On the other hand, the latter serves well in detecting anomalies that deviate from the normal behaviour of the system whether this abnormality was encountered before or not. Anomaly-based

IDS mostly depend on using statistical modeling to define a system profile reflecting its normal behaviour. This approach ignores the fact that the normal behavior of a system changes over time, unless the model is frequently updated/recreated. Other anomaly-based approaches include predicting future behaviour of the system given its history. Although these approaches are more successful in capturing changes in the normal behavior of a system, they require a longer training period and, in some cases, their application can be infeasible because of the size of data sets involved.

After an IDS is developed, its performance needs to be evaluated. There are two approaches to evaluate an IDS: online evaluation and offline evaluation. Online evaluation requires a real physical network to be built, where network traffic (both background and attack) is generated by real users. The IDS is then deployed in the network so that its capability of detecting malicious activity (i.e. intrusions) can be put under test through measuring the rate of correct identification of events, in addition to the rate of false alarms. On the other hand, offline evaluation only requires the existence of a network traffic trace (i.e. a network dataset) that represents the traffic that the IDS will most probably encounter in the network it is supposed to protect. The dataset can be generated by simulating the network the IDS will be deployed in or a publicly available trace can be used. Publicly available traces can either be anonymized traces of real network traffic, or traffic that has been generated in a way to resemble real network traffic. The dataset can then be replayed against the IDS and the performance of the IDS will be measured based on both the rate of correct identification of events and the rate of false alarms.

## 1.2 Challenges

While IDSes slowly evolved in the past few years, some challenges still lie ahead for the coming generation of IDSes. Animesh Patcha and Jung-Min Park [96] summarize those challenges as a conclusion to their survey of various anomaly-based IDS techniques. First and foremost, next generation IDSes should adapt to the evolving networking paradigms like wireless, mobile networks, and high speed networks. In addition, the ability of an IDS to adequately detect future intrusions has to be enhanced. Not only should an IDS be able to identify novel attacks, but it should also have a low rate of false alarms. A false alarm is caused by misidentifying the nature (normal or abnormal) of the network data under test. [32] defines the ideal minimum rate of false alarms in an IDS to be 1 false alarm per 100,000 events. However, there is currently a challenge with having a general systematic methodology and/or a set of metrics to fairly evaluate different IDS algorithms. This leads to the problem addressed in this thesis: the lack of an appropriate IDS evaluation dataset that fits the common research needs. Some researchers tend to use anonymized versions of private datasets extracted from real life environments to evaluate

their own IDS algorithms. This raises the question of how can private entities provide the anonymized versions of their own network traffic traces while preserving their users' data security and privacy without affecting the evaluation process (the detection accuracy in specific) of the IDS algorithm under test. Modeling the normal behavior of a system in order to correctly detect intrusions is still considered an on-going research problem as well. Finally, an insider threat study [70] done at the Software Engineering Institute, Carnegie Mellon found that 29% of attacks are caused by insiders. This creates another challenging problem to be solved in the field of IDS which is detecting internal attacks.

## 1.3 Motivation

Despite facing many challenges in the research field of IDS, IDS evaluation (both online and offline) remains an open research problem. No matter which problem one tries to solve in the field of IDS or which feature one tries to add to an existing IDS, there will always be a phase where one has to evaluate her newly devised technique, system or algorithm. However, the main focus of previous research efforts in the field of IDS was to come up with new ways of detecting intrusions and enhancing current techniques to achieve better performance. This left the research community with very little support for IDS evaluation, most of which did not satisfy most of the common research needs. This triggered us to survey those needs and come up with a set of design requirements for a system that satisfies as much as possible of such needs. This led us to build an open-source configurable virtual testbed for to generate datasets for offline evaluation of IDSes, Algorizmi.

## 1.4 Contributions

The contributions of this thesis are twofold. First, we provide an overview of the challenges and open problems in the field of IDSes. The challenges we face in the field mainly revolve around devising new or improving current IDS algorithms. However, one of those open problems remains orthogonal to all other problems. That is how we can provide the research community with a system that will help generate network traffic traces (network datasets) that can then be used for IDS offline evaluation.

To solve this problem, we surveyed previous efforts done by various researchers and extracted the needs of the community in a system that will allow them to evaluate different IDS algorithms. We then provide a list of design requirements in order to build a system that will satisfy those research needs.

We, then, propose an open-source configurable virtual testbed (Algorizmi) that provides its users with tools that allow them to generate datasets useful for offline evaluation of

IDSes. Each user can have more than one experiment running on Algorizmi at any given time. An experiment is in fact a virtual network created on the testbed. A user will be capable of choosing and configuring the desired virtual machine instances to run on any of her given experiments. In addition, each user has the ability to create SSH key pairs in case they want to access an instance from outside Algorizmi. In order to facilitate background traffic generation, we provide Algorizmi users with background script templates that they can make use of to generate their own background scripts that will run within their experiments. Since generating attacks is a key requirement for Algorizmi, users will be provided with attack traffic generation tools to help them launch attacks against virtual machine instances running within their experiments. A user can generate the dataset for any of her experiments at any given time. Finally, she can use that dataset to evaluate her IDS in an offline fashion the same way researchers use the DARPA dataset.

## 1.5 Thesis Organization

In this thesis, we present Algorizmi, an open-source configurable virtual testbed to generate datasets for offline evaluation of IDSes. Algorizmi makes use of the advantages of previous efforts done in IDS evaluation and avoids discovered caveats. The rest of this thesis is organized as follows. Chapter 2 provides a review of the work done in the field of IDS evaluation. The design requirements we extracted from the research needs per our literature review and on which we based Algorizmi are given in Chapter 3. We also show how related work complies to those requirements. Chapter 4 illustrates the architecture of Algorizmi, a closer look at its system components, and some of its common use cases. Chapter 5 shows the methodology adopted to evaluate Algorizmi and the results of our evaluation. In Chapter 6, we conclude this thesis by mentioning lessons learned from building Algorizmi and our suggestions for future work.



# Chapter 2

## Literature Review

### 2.1 Introduction

An important aspect of IDS research is providing evaluation mechanisms to assess IDS techniques developed by various researchers. Previous efforts involved offering private data traces after anonymizing sensitive user information as well as providing public data traces. A researcher can then replay a data trace by feeding it into an IDS to collect generated alerts. Generally, the performance of an IDS is then measured by the number of attacks it successfully discovered and the number of missed attacks. The DARPA datasets publicly available from MIT Lincoln Labs [57] are the most significant contribution in that area and have been extensively used within the IDS research community. However, the process of generating those datasets was heavily criticized [86]. Frédéric et al. present another dataset generation tool [85] that offers a configurable testbed based on virtualization concepts. Nevertheless, their system is limited to evaluating signature-based network IDS. Other researchers [111, 90, 72, 37] offered anonymized data traces after removing sensitive user information. The major disadvantage of private traces is the tradeoff between data usability and user privacy. The higher the user privacy level is, the less usable the obtained data trace is.

Other research efforts provided platforms that can be used for IDS evaluation. However, such platforms are either obsolete [87, 48], not available to the community [107] or very generic to be used for IDS evaluation [35, 93]. Despite the efforts in IDS research, evaluation datasets that simulate realistic network environments enabling fair comparison of various IDS techniques are still missing. There is still a critical need to build more appropriate evaluation datasets [96].

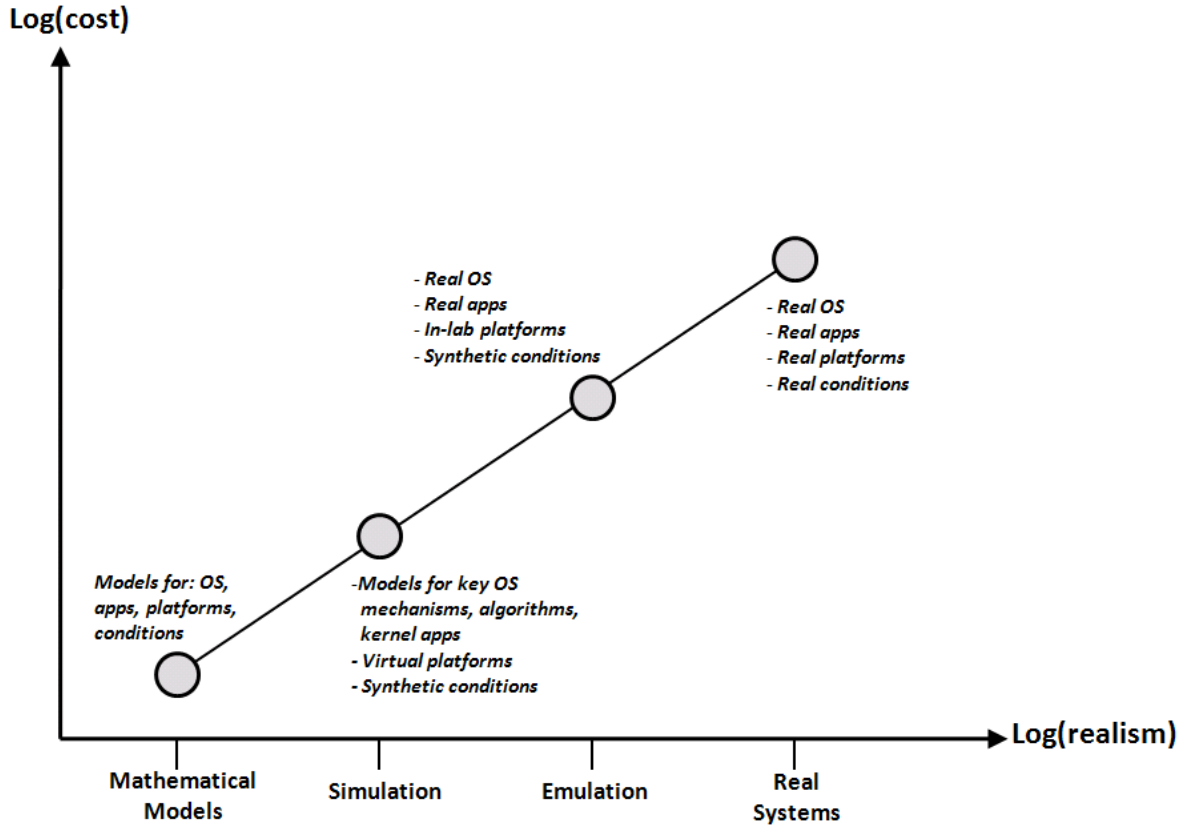


Figure 2.1: Four approaches of network experiments [93]

## 2.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 2.3 provides an overview of the different approaches of performing network experiments. Section 2.4 briefly discusses different ways to evaluate an IDS algorithm. In this chapter, we review previous IDS evaluation efforts that used anonymized datasets. We then offer an overview of various IDS evaluation projects that used synthesized traffic in a controlled environment. Finally, Section 2.5 provides a summary of the related work presented in this chapter.

## 2.3 Network Experiments

Generally, there are four main approaches that can be used to perform network experiments [93]. These include mathematical models, simulation, emulation and using real systems.

Section 2.1 compares these four approaches in terms of the tradeoff between their cost (monetary cost and time cost) and their level of realism.

### 2.3.1 Mathematical Models

Mathematical models are the most theoretical choice and are usually the first step of any research. In such models, the given system, applications, platforms and conditions are modeled (e.g., stochastic models, non-linear models) then validated mathematically.

### 2.3.2 Simulation

Evaluating IDSes by simulating real network entities is a first step towards practical evaluation where models for key operating system mechanisms, algorithms and kernel applications, virtual platforms, networks, and synthetic conditions are developed. However, it is difficult to build a realistic simulation of the Internet [53]. Examples of such simulators include SimGrid [42], MicroGrid [115], Bricks [118] and NS [52].

### 2.3.3 Emulation

A more realistic tool used for evaluation is emulation where real operating systems, real applications, in-lab platforms, networks and synthetic conditions are used. This allows researchers to control the experiment conditions and to reproduce the results by applying the same conditions in the future. However, it still lacks the practical conditions that are present in a real network. WAN-in-Lab (WIL) [79] and Emulab [125] are good examples of testbeds that try to emulate real systems.

### 2.3.4 Real System

Using a real system means using real operating systems, real applications, real platforms, real networks and real conditions. The main problem is that researchers are unable to control the background traffic and, more importantly, unable to reproduce the same experiment conditions. In addition, the process of IDS evaluation involves running network attacks against the machines (or the network) the IDS under test is protecting, which may render the whole network inoperable when those attacks are successful. This will incur more time and effort to restart the experiment when required. Therefore, emulation seems more appropriate than using a real environment for IDS evaluation. Examples of real experimental testbeds include Grid'5000 [41], TERAGrid [43] and PlanetLab [45].

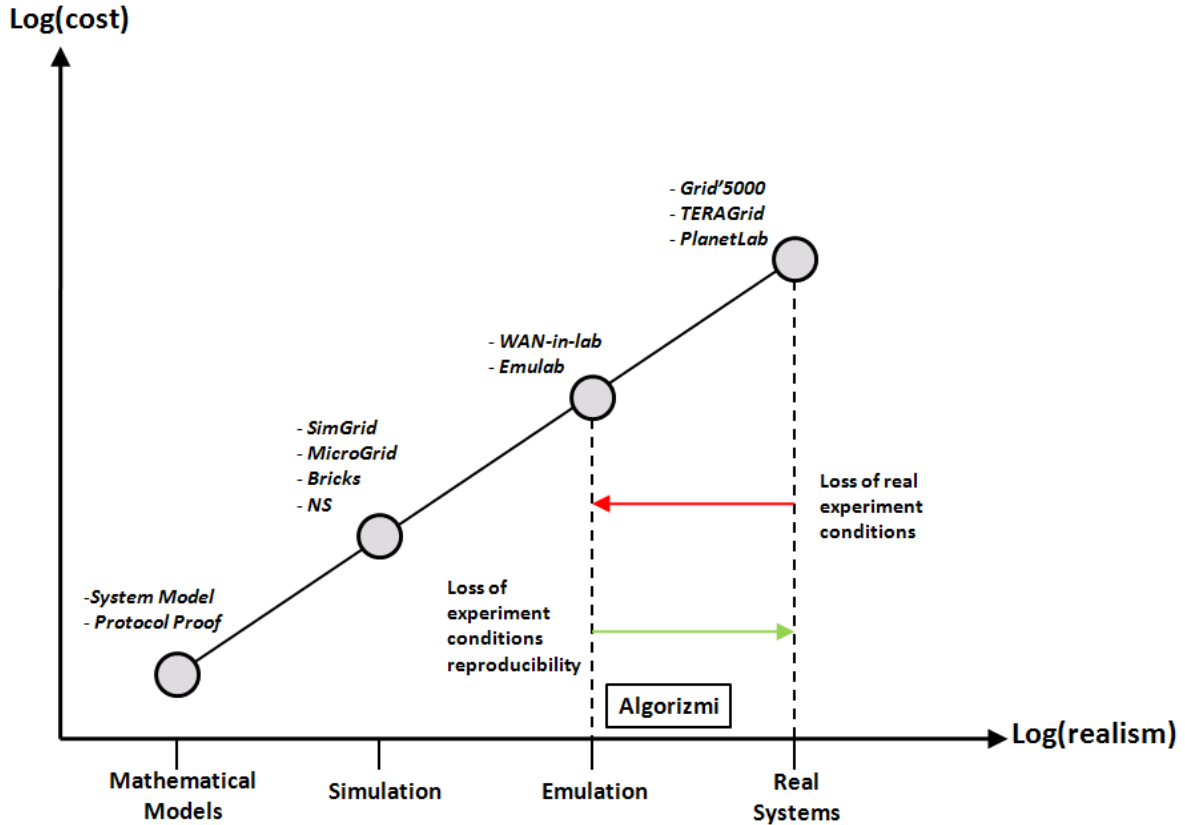


Figure 2.2: Examples of experimental tools corresponding to the four approaches of network experiments [93]

## 2.4 Offline Evaluation of Intrusion Detection Systems

Traditional IDS research did not focus much on providing reliable evaluation datasets that can be used for fair comparisons between various IDS techniques. Athanasiades et al. [31] describe the process undertaken by most of the previous efforts in IDS testing and evaluation as an ad-hoc methodology making it difficult to compare different algorithms and approaches. However, there were many efforts done to overcome this problem. In the literature of IDS evaluation, researchers usually used either private data traces after anonymizing sensitive information [111, 72] or generated data traces out of synthesized traffic in a controlled environment [57] in order to evaluate a newly devised IDS algorithm.

### 2.4.1 Dataset Anonymization

Several researchers used real traffic traces after anonymizing sensitive data [111, 90, 72, 37]. However, there is always a tradeoff between the level of data privacy and the utility of the released dataset. Moreover, there are many controversial legal issues concerning monitoring and collecting real traffic traces [29, 112]. Besides, such datasets throw away the packet payload since it is a very sensitive aspect of the collected traces. Therefore, it is impossible to use such datasets for some types of IDS algorithms that need to look into the packet payload (e.g. worm detectors, string matching-based techniques).

### 2.4.2 Dataset Generation

This section presents previous research efforts that used dataset generation in order to evaluate IDSes.

#### UCD Platform

This platform was developed in the University of California at Davis (UCD). It is considered the first IDS testing platform that automatically launches attacks using interactive telnet, FTP and rlogin sessions [87]. Scripts were used to generate background traffic and attack traffic. The ability of the IDS under test to distinguish between intrusions and normal behavior was then evaluated [103]. Network Security Monitor (NSM) [60] was one of the earliest IDSes to be evaluated using this platform.

#### IBM Zurich IDS Testing Platform

The Network Security and Cryptography group at IBM Zurich Research Laboratory developed a similar testing platform to support IDS research [48]. Both background and attack data were generated as in [103]. However, background traffic was generated only for FTP servers. Moreover, very few FTP attacks were simulated in the attack data.

#### DARPA Datasets

This is the earliest effort by Lincoln Laboratory at Massachusetts Institute of Technology (MIT/LL) to assess the performance of various IDS techniques. The DARPA 1998 and 1999 intrusion detection evaluations [57] are the first systematic effort towards IDS evaluation [31]. MIT/LL researchers had to overcome the problem of the non-existence of standard

comparison metrics, attacks, background traffic or methodology [57]. Being publicly available, both datasets have been used extensively by researchers to evaluate various types of IDS algorithms: classification-based [81], clustering-based [129, 140, 116, 131], immunity-based [144], neural networks approaches [59, 142, 134], machine learning approaches [68] and statistical-based [44, 67, 50]. However, the DARPA data generation techniques were heavily criticized [86]. The main points of criticism are as follows:

### 1. Background Data:

- The process used to generate the background noise is only superficially described in the reports.
- The methodology for proving that the statistics of the generated traffic match that of the real traffic was never published.
- It is unknown how many false alarms the background traffic might generate.
- Data rates were never variable.
- Some inconsistencies concerning the network setup of the testbed were found among various documentation reports produced by DARPA.

### 2. Attack Data:

- Synthetic attacks were unrealistically evenly distributed throughout the background traffic.
- Each attack type was used the same number of times reflecting an unrealistic behavior.
- Only a subset of the systems was subject to interactive attacks.
- The unrealistic architecture of the used testbed is only implicitly acknowledged in [73].
- The database of attacks used is now obsolete and is not extensible. Therefore, it is impossible to add new attacks to it now or in the future.

## Lincoln Adaptable Real-Time Information Assurance Testbed (LARIAT)

MIT/LL developed LARIAT to extend its previous efforts in the field of IDS evaluation. LARIAT is a set of tools designed to assist in the evaluation and configuration of information assurance technologies [107]. It came after the DARPA 1999 evaluation with its first version released in 2000, followed by another extension in 2001. LARIAT offered background traffic profiles as well as profiles for attack data. However, the traffic data used is similar to that used by the DARPA 1999 evaluation which was already criticized

[86]. Moreover, the system was designed to run one attack scenario at a time where a clean up phase is run after each attack to clear any traces of the attack and restore the environment to its previous state. This is not useful when the attack consists of multiple phases and each is considered an attack by itself. For example, portscan/probe usually precedes most types of attacks (denial of service, buffer overflow, worms). If the side-effects of portscanning are cleared, the IDS might not be able to detect the attack afterwards. A more important disadvantage of LARIAT is that it is not publicly available to the research community.

## Cyber DEfense Technology Experimental Research (DETER) Network

DETER is a collaborative work between USC Information Sciences Institute, University of California at Berkeley and McAfee Research [35]. The DETER testbed is intended to provide an experimental infrastructure to support the development and demonstration of next-generation information security technologies [34]. It is implemented as an Emulab cluster [126]. Although DETER offers a reliable testbed for experiments that involve malicious code and for experimenting DoS and DDoS defense mechanisms, it was not designed to be a testbed for evaluating IDS algorithms. This explains why none of the projects hosted on DETER <sup>1</sup> is related to IDS evaluation.

## Virtual Network-based IDS Evaluation

Frédéric et al. propose a virtual network infrastructure for IDS evaluation [85] that uses virtual machines (through VMware) to build a controlled testbed. The system has five main steps for traffic generation:

1. **Script Generation:** choose which exploit will be run against a target system.
2. **Virtual Network Setup:** build a different virtual network for each attack script.
3. **Current Attack Script Setup:** configure the current attack scenario.
4. **Attack Execution:** launch the attack while recording the generated traffic.
5. **Tear Down:** save traffic traces on a shared drive, then restore the virtual machines to their initial states.

Although this environment offers many advantages that are not present in previous work done on IDS evaluation, there are several problems in the proposed system:

---

<sup>1</sup><http://www.isi.deterlab.net/projectlist.php3>

1. The authors acknowledge that the generated dataset is specific to signature-based network IDS. Consequently, the system is not useful for any kind of anomaly-based IDses which are more dominant in the research field [87].
2. No background traffic is generated which raises questions about the given false alarm rates in their evaluation.
3. Attack scenarios are run one at a time sharing the same disadvantage with LARIAT.
4. No detailed information is given on how to obtain this system and use it.

### **LAAS Network Experiments (LassNetExp)**

LaasNetExp [93] is a generic polymorphic platform for network emulation and experiments developed by Laboratory for Analysis and Architecture of Systems (LAAS) in France. It is among the most recent contributions in the field of network experimentation. The system provides:

1. Reproducible experiments by means of controlled traffic.
2. Measurement and monitoring tools.
3. Isolation from real world.
4. Platform management and configuration.

However, LaasNetExp is not designed for IDS evaluation so there are no tools to generate synthesized attack traffic or emulate attack machines. In addition, no details are given on how this platform is available to the research community.

## **2.5 Summary**

In this chapter, we gave an overview of the four approaches commonly used to do network experiments: modeling, simulation, emulation and real systems. We also provided a review of the projects done based on the two common methods to evaluate an IDS: anonymizing real network traffic traces and generated synthesized network traffic traces. Table 2.1 summarizes the previous research work presented in this chapter.



<b>Project</b>	<b>Nature</b>	<b>References</b>
Privacy-preserving Anomaly Detection for Network Traffic	Dataset Anonymization	[37]
SANTT	Dataset Anonymization	[111]
SC2D	Dataset Anonymization Alternative	[90]
A survey of state-of-the-art in anonymity metrics	Dataset Anonymization Metrics	[72]
UCD Platform	IDS Testing Platform	[87]
IBM Zurich IDS Testing Platform	IDS Testing Platform	[48]
DARPA Datasets	Dataset Generation	[57]
LARIAT	IDS Testing Platform	[107]
DETER	Experimental Testbed	[35]
Virtual Network-based IDS Evaluation	Virtual Network Infrastructure	[85]
LAASNETEXP	Network Emulation Platform	[93]

Table 2.1: Summary of related projects

# Chapter 3

## Design Requirements

### 3.1 Introduction

Building a configurable virtual testbed to generate datasets for offline IDS evaluation imposes some design requirements on the proposed platform in order to have the maximum benefit out of it. We surveyed all the previous systems that generated datasets for offline IDS evaluation and identified the design requirements of such a system. We then based the design of Algorizmi on those requirements. Each category of design requirements is followed by a table that evaluates the systems discussed in Chapter 2 based on whether the design requirements are satisfied or not.

### 3.2 Chapter Organization

The rest of this chapter is organized as follows. Section 3.3 discusses the availability requirements that should be present in a configurable virtual testbed for IDS evaluation, focusing on Algorizmi in particular. Versatility requirements are explained in Section 3.4. Section 3.5 highlights the design requirements needed to make both the generated background traffic and attack traffic similar to real network traffic. Section 3.6 and Section 3.7 show how we can efficiently build such a system with minimum cost. Section 3.8 discusses the functional requirements. Section 3.9 provides the design requirements for containment and security. Section 3.10 discusses usability requirements which drive the user friendliness of Algorizmi. Section 3.11 discusses the design requirements that will enable Algorizmi users to reproduce their experiments. Following, Section 3.12 highlights the design requirements imposed on maintaining Algorizmi. Finally, Section 3.13 concludes the chapter by providing an overview of the design requirements discussed and how each previous research work succeeds in/fails to satisfy those requirements.

### 3.3 Availability

An IDS evaluation dataset generation tool (Algorizmi in specific) should be publicly available to the research community to achieve maximum benefit by allowing interaction among users, and sharing the information that users would like to make public (e.g virtual machine images, background scripts, attack scripts and datasets). Moreover, the source code itself should be open to the community for public scrutiny and to allow the community to contribute to the code in order to enhance Algorizmi and add more features to it. As shown in Section 3.1, only four of the previous research projects are publicly available while none of them is open source.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Publicly available	✓	-	-	✓	-	✓	✓	-
Open source	-	-	-	-	-	-	-	-

Table 3.1: Compliance of previous research efforts to the Availability design requirements

### 3.4 Versatility

The user should have the choice of deploying virtual machines of several types and should be able to select the configuration for memory, operating system, and instance storage that suit the needs of her experiment. With the recent advancements in networking, the user might want to emulate a high bandwidth network. Algorizmi should provide the user with the appropriate tools to achieve this design. In addition, the user might want to model various types of networks; for example, hierarchical networks, flat networks. This should also be supported by Algorizmi. Table 3.2 shows that there is only one project that offers a configurable environment and different network models for its users.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Configurable environment	-	-	-	-	✓	-	✓	✓
High-bandwidth network emulation	-	-	-	-	-	-	-	✓
Different network models	-	-	-	-	✓	✓	✓	-

Table 3.2: Compliance of previous research efforts to the Versatility design requirements

## 3.5 Fidelity

Although Algorizmi is an emulation environment, it should provide the users with tunable experimental conditions, that are as close to real environment conditions as possible, in two major aspects: background traffic generation and attack traffic generation.

### 3.5.1 Background Traffic

Firstly, the user should have the ability to tune and control the background traffic properties. This will allow controlling both the load of the network and host machines. Having such control will help reproduce experimental results if the same settings are reused. Secondly, Algorizmi should offer the user with pre-set models to facilitate background traffic generation. Thirdly, the user should be able to add her own models and apply them to her experimental testbed. Algorizmi should also provide the user with the necessary tools required to extract such models from her own traffic traces keeping in mind that such traffic traces can be real traces. Consequently, the process of extracting the model must not gather any kind of sensitive information from the traces. Finally, new models by the user will be added to pre-existing ones in order to keep a rich database of background traffic generation models that will be shared among the research community. Table 3.3 shows that LARIAT [58] fully satisfies all the design requirements of generating background traffic that is as close as possible to real-life network traffic. Other research efforts, however, only partially satisfy the requirements or none at all.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Tuning	-	-	-	-	✓	✓	✓	✓
Pre-set models	-	-	-	-	✓	-	-	-
Add new models	-	-	-	-	✓	✓	-	✓
Extract model from available traces	-	-	-	-	✓	-	✓	✓
Models database	-	-	-	-	✓	-	✓	✓

Table 3.3: Compliance of previous research efforts to the Background Traffic Fidelity design requirements

### 3.5.2 Attack Traffic

Firstly, Algorizmi should allow the user to select which attacks will be launched against victim machines such that only those attacks of interest will be present in the final dataset. Since Algorizmi can be used to evaluate both signature-based and anomaly-based IDSes, the user should be able to define when the attack phase will start to allow for a portion of

the generated dataset to be free of attacks. This will be useful in training anomaly-based algorithms. Moreover, there should be a way to frequently update the database of attack scripts used to launch attacks against the virtual testbed. In addition, the user might want to use her own attack scripts. In such cases, the attack scripts database should be designed to accommodate new attack scripts if the user wants to share them with other users. Finally, varying the characteristics of the attacking machines so that it will not favor an anomaly-based algorithm over another should also be taken into consideration [87]. Table 3.4 shows that none of the projects varies the attacking machines, and only LARIAT [58] allows its users to select the attacks in their experiments from an up-to-date database of attacks. However, as perviously mentioned LARIAT is not open for public use.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Attack selection	-	-	-	-	✓	✓	✓	-
Attack-free period	-	-	-	✓	✓	-	-	-
Updatable database	-	-	-	-	✓	-	-	-
Different attacking machines	-	-	-	-	-	-	-	-

Table 3.4: Compliance of previous research efforts to the Attack Traffic Fidelity design requirements

### 3.6 Economy

Such a research testbed should not depend on expensive hardware or software, but rather on commodity hardware and open source software components. This will have a direct effect on reducing the development and maintenance costs. Table 3.5 shows that none of the previous research projects gives exact or approximate figures for the cost of implementation.

### 3.7 Efficiency

Since Algorizmi will be shared among a potentially large number of users, the underlying physical resources should be efficiently used so as to maximize resource utilization. Table 3.5 shows that only two projects (DETER [35] and LaasNetExp [93]) provide details on how the available resources are fully utilized by their system.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Low cost	-	-	-	-	-	-	-	-
Resource utilization	-	-	-	-	-	✓	-	✓

Table 3.5: Compliance of previous research efforts to the Economy and Efficiency design requirements

## 3.8 Functionality

A set of measurement and data collection tools should be included so that Algorizmi will function properly as an emulation platform for IDS evaluation.

### 3.8.1 Measurement Tools

Measurement and monitoring tools should be incorporated into Algorizmi to collect data traces that will build up the dataset generated from an experiment. It is important to ensure that running any of the measurement tools should not disrupt the experiment and should provide log files without missing any event. Table 3.6 shows that all projects provided trace collection tools that are used to eventually generate the datasets of the experiments.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Trace collection tools	✓	✓	✓	✓	✓	✓	✓	✓
Non-disruptive running of tools	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.6: Compliance of previous research efforts to the Measurement Tools design requirements

### 3.8.2 Datasets

Firstly, sharing the generated dataset of a virtual testbed should be allowed if the testbed owner chooses to share it so that other users of Algorizmi can later make use of the generated datasets. It is important to note that proper documentation of the generated dataset is essential to achieve better usage. It is also important to know key information (target OS, service, attack specification) about the attacks present in a generated dataset. Eventually, this will help automating the process of IDS evaluation. In addition, having different output formats for the generated dataset will facilitate the process of IDS evaluation for many researchers. Some IDS algorithms (e.g. statistical-based) require the data

to be represented in formats other than raw packet format (e.g, number-based format). Table 3.7 shows that anonymized datasets, DARPA datasets and LaasNetExp dataset are the only datasets that are shared publicly in the community and available to use. Despite the fact that all the generated datasets are documented, none is generated in different output formats.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Shared datasets	✓	-	-	✓	-	-	-	✓
Documented datasets	✓	✓	✓	✓	✓	✓	✓	✓
Different Output formats	-	-	-	-	-	-	-	-

Table 3.7: Compliance of previous research efforts to the Dataset design requirements

### 3.9 Containment and Security

Since Algorizmi offers a controlled environment for each experiment, it should be isolated from the real network to protect it against attackers. In other words, the virtual machines in a testbed are not accessible from outside Algorizmi except through an SSH key pair provided the system. Additionally, Algorizmi users are primarily researchers who want to make sure their algorithms or experiment results are not leaked to the public before having them published. Therefore, the platform should protect the data of an experiment unless the user wants to publicly share it. Although four projects isolate the testbeds used to generate the datasets from the Internet, Table 3.8 shows that only one project (DETER [35]) explains how it protects private user data from being exposed to other users.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Isolation from the Internet	-	-	-	-	✓	✓	✓	✓
Data protection	-	-	-	-	-	✓	-	-

Table 3.8: Compliance of previous research efforts to the Containment and Security design requirements

### 3.10 Usability

Algorizmi should provide the user with an easy way to smoothly deploy the virtual testbed with minimum effort. The user should also be given the option to reconfigure the testbed

to perform other experiments. Not only should a user have full control of the entities constituting her experimental testbed, but she is also able to scale the testbed up/down in a seamless way. Interaction with Algorizmi should be offered in an easy-to-use way such that the learning curve of knowing how to use the system is adequate. Table 3.9 shows that four projects satisfy most of the usability design requirements.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Smooth deployment	-	-	-	-	✓	✓	✓	✓
Testbed reconfiguration	-	-	-	-	✓	✓	✓	✓
Full control on testbed	-	-	-	-	✓	✓	✓	✓
Scalable testbed	-	-	-	-	✓	✓	✓	✓
User friendly interface	-	-	-	-	-	✓	-	-

Table 3.9: Compliance of previous research efforts to the Usability design requirements

### 3.11 Reproducibility

The user should be able to save the initial configuration of an experiment so that the testbed can be reset whenever required. In addition, it is useful to offer the user a way to save the configuration of an experiment if it has been modified since starting it. Various experiment configurations can be stored on a database so that other users can make use of them afterwards, in case the owner of the experiment wishes to share this configuration.

### 3.12 Maintenance

Algorizmi should be flexible enough to accommodate replacing faulty resources or adding new ones to the physical infrastructure without disrupting any running experiment. Table 3.10 shows that only two projects (DETER [35] and LaasNetExp [93]) fully satisfy the reproducibility and maintenance design requirements.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
Save experiment configuration	-	-	-	-	✓	✓	✓	✓
Resource replacement	-	-	-	-	-	✓	-	✓

Table 3.10: Compliance of previous research efforts to the Reproducibility and Maintenance design requirements



### 3.13 Summary

In this chapter, we have presented an overview of the design requirements identified by surveying the literature. We analyzed the compliance of each research project discussed in Chapter 2 with each design requirement. The final result of our analysis shows that MIT/LL LARIAT [58], DETER [35] and LassNetExp [93] satisfy the highest subset of requirements among all the projects. Although the DARPA dataset satisfies a very small subset of the requirements, it has been widely used by the research community ever since it was made available in 1998. This is mainly attributed to the fact that the DARPA dataset is a static dataset, i.e users do not have the ability to change any configuration in the dataset. In addition, users cannot select or tune the attacks presented in the DARPA dataset. Table 3.11 summarizes our analysis.

	Anonymized Datasets	UCD	IBM Zurich	DARPA	LARIAT	DETER	Frédéric et al.	LaasNetExp
<b>Availability</b>								
Publicly available	✓	-	-	✓	-	✓	✓	-
Open source	-	-	-	-	-	-	-	-
<b>Versatility</b>								
Configurable environment	-	-	-	-	✓	-	✓	✓
High-bandwidth network emulation	-	-	-	-	-	-	-	✓
Different network models	-	-	-	-	✓	✓	✓	-
<b>Background Traffic Fidelity</b>								
Tuning	-	-	-	-	✓	✓	✓	✓
Pre-set models	-	-	-	-	✓	-	-	-
Add new models	-	-	-	-	✓	✓	-	✓
Extract model from available traces	-	-	-	-	✓	-	✓	✓
Models database	-	-	-	-	✓	-	✓	✓
<b>Attack Traffic Fidelity</b>								
Attack selection	-	-	-	-	✓	✓	✓	-
Attack-free period	-	-	-	✓	✓	-	-	-
Updateable database	-	-	-	-	✓	-	-	-
Different attacking machines	-	-	-	-	-	-	-	-
<b>Economy and Efficiency</b>								
Low cost	-	-	-	-	-	-	-	-
Resource utilization	-	-	-	-	-	✓	-	✓
<b>Measurement Tools</b>								
Trace collection tools	✓	✓	✓	✓	✓	✓	✓	✓
Non-disruptive running of tools	✓	✓	✓	✓	✓	✓	✓	✓
<b>Datasets</b>								
Shared datasets	✓	-	-	✓	-	-	-	✓
Documented datasets	✓	✓	✓	✓	✓	✓	✓	✓
Different Output formats	-	-	-	-	-	-	-	-
<b>Containment and Security</b>								
Isolation from the Internet	-	-	-	-	✓	✓	✓	✓
Data protection	-	-	-	-	-	✓	-	-
<b>Usability</b>								
Smooth deployment	-	-	-	-	✓	✓	✓	✓
Testbed reconfiguration	-	-	-	-	✓	✓	✓	✓
Full control on testbed	-	-	-	-	✓	✓	✓	✓
Scalable testbed	-	-	-	-	✓	✓	✓	✓
User friendly interface	-	-	-	-	-	✓	-	-
<b>Reproducibility and Maintenance</b>								
Save experiment configuration	-	-	-	-	✓	✓	✓	✓
Resource replacement	-	-	-	-	-	✓	-	✓

Table 3.11: Compliance of previous research efforts in the field of IDS evaluation to our identified design requirements

# Chapter 4

## Algorizmi

### 4.1 Introduction

Algorizmi is built as a tool that helps users generate network datasets (i.e. network traffic traces) that can then be used for IDS offline evaluation. The concept of virtualization is central to the architecture of Algorizmi. In other words, Algorizmi makes use of machine virtualization to allow its users to build several network, each using a defined set of physical nodes. Users can then configure the traffic (both background and attack) generated in their networks. At any given time, a user will be able to download the dataset that represents her experiment to use it afterwards to perform an offline evaluation of her IDS.

### 4.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 4.3 provides an architectural overview of Algorizmi. A detailed explanation for the components that build up the design of our system is given in Section 4.4. Sample use cases for Algorizmi are given in Section 4.5 to show how it can be used by the research community. We conclude the chapter in Section 4.6.

### 4.3 Architectural Overview

In order to satisfy the requirements detailed in Chapter 3, Algorizmi represents each experiment as a virtual network hosted on physical machines. Each virtual network has its own resources so as not to conflict with other experiments. Users of a specific experiment

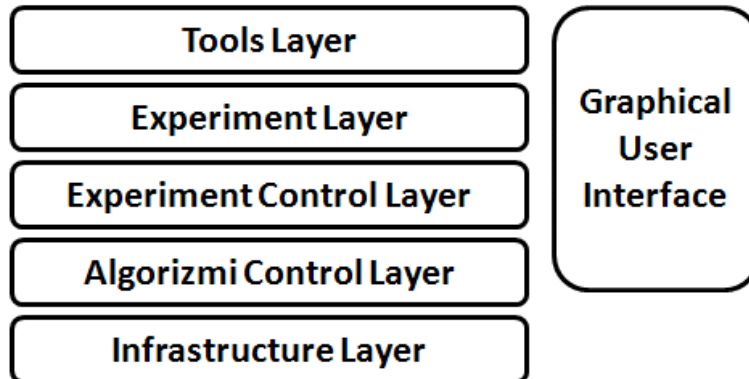


Figure 4.1: Algorizmi Abstract Architecture

will be able to configure it, select which Virtual Machine Images (VMIs) to load, configure and save. Moreover, they will be able to tune how background traffic will be generated, as well as which attacks will be launched against target machines. They will have the option of running selected tools on top of their experiment; for example, to collect data traces that will eventually contribute to generating the experiment dataset. Figure 4.1 gives an architectural overview of Algorizmi.

### 4.3.1 Infrastructure Layer

The infrastructure layer represents the physical infrastructure of Algorizmi. This layer includes two types of nodes:

1. **Storage Nodes:** responsible for storing the generated datasets, experiment configurations and other settings
2. **Physical Machines:** that will act as hosts for the virtual machines contributing to the various virtual testbeds.

In addition, network resources connecting various physical host machine and storage nodes are included in this layer. The infrastructure layer of Algorizmi is shared among all experiments running on the platform.

### 4.3.2 Algorizmi Control Layer

The Algorizmi Control layer controls the operation of all the nodes in Algorizmi. It is responsible for managing the experiment setup in two aspects: allocating physical nodes

required for running the virtual environment of an experiment and allocating the physical storage associated with an experiment. The functions of this layer are:

1. Ensuring the safe operation of all nodes in Algorizmi.
2. Managing the process of initiating new experiments.
3. Allocating resources for various experiments running on Algorizmi per user request.
4. Monitoring access to the platform.
5. Observing appropriate user access rights to any data present within the platform.
6. Overriding the Experiment Control layer in the case of an out-of-control experiment.
7. Maintaining various databases used within the platform: attacks database, experiments configuration database, datasets database and virtual images database.

### **4.3.3 Experiment Control Layer**

The main functionality of the Experiment Control layer is to hand the control of a specific experiment over to its users. This includes:

1. Providing users with access to their pre-allocated resources.
2. Loading and configuring user-specified virtual machine instances.
3. Configuring experiment settings: background traffic generation, attack traffic generation and network setup.
4. Loading any other experiment-specific data and settings.
5. Saving the current state of the experiment for later uses.
6. Monitoring the operation of the virtual nodes within the experiment.
7. Compiling data traces to eventually generate the experiment dataset.
8. Restoring the experiment to its initial state if required.
9. Terminating the experiment.

### 4.3.4 Experiment Layer

This layer represents the emulated testbed of an experiment. Its scope includes all the nodes involved in that specific experiment, allowing unlimited interactions among them. This layer includes:

1. Normal virtual machine instances.
2. Attacking virtual machine instances.
3. Background and attack traffic scripts.
4. Virtual networks connecting the experiment's virtual machine instances.

### 4.3.5 Tools Layer

The Tools layer is flexible as it includes any tool the user requests to run on her experimental testbed. Currently Algorizmi offers two sets of tools:

1. **Data Collection Tools:** required to log data traces for the dataset compilation process. An examples of a tool in this category is: tcpdump [21].
2. **Attack Verification Tools:** required to verify the success of attacks launched against guest machines. Metasploit Framework (MSF) [11] is a good example of an attack verification tool.

Other tools can be added to this layer by Algorizmi administrators or by the research community.

### 4.3.6 Graphical User Interface

The Graphical User Interface layer offers an easy-to-use interface for the users of Algorizmi to access their resources, each according to her granted rights. It is a Java-based desktop application and hence can be used on multiple platforms without the need to change or recompile the code.

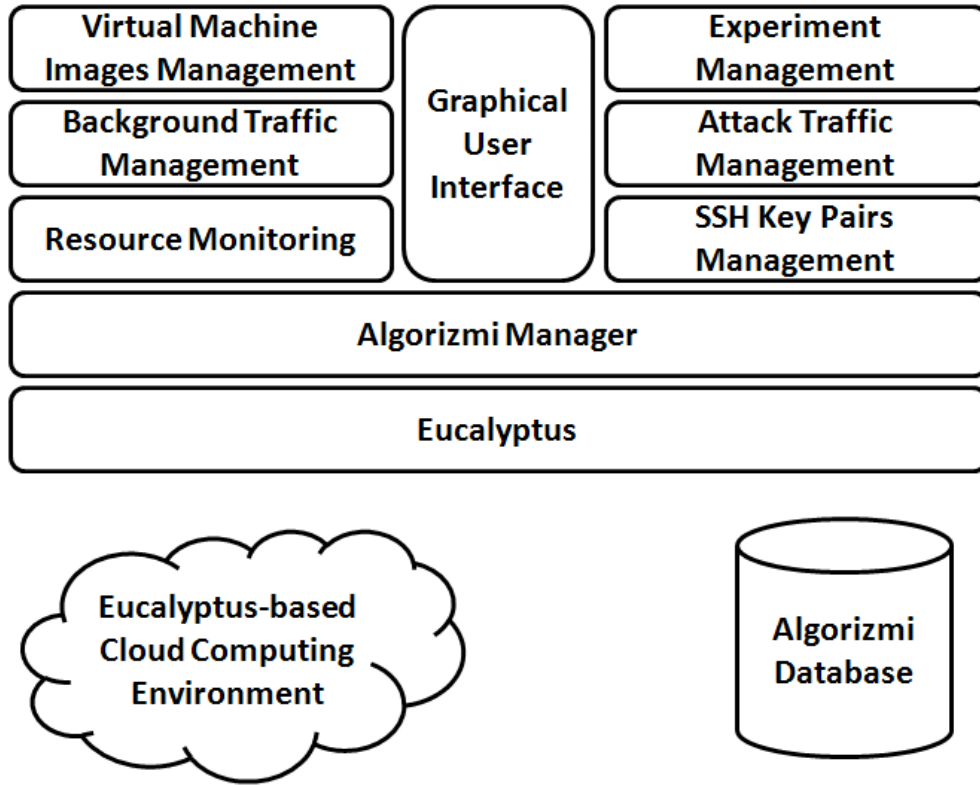


Figure 4.2: Components of Algorizmi

## 4.4 System Components

The basic components that contribute to the infrastructure of Algorizmi are illustrated in Figure 4.2. There are four basic layers of components:

1. Infrastructure components
2. Eucalyptus
3. Algorizmi Manager
4. Various application components

### 4.4.1 Infrastructure Components

The infrastructure layer that is shared among all Algorizmi users is based on two components: a Eucalyptus-based cloud computing environment [91] and Algorizmi database.

#### Eucalyptus-based Cloud Computing Environment

Eucalyptus is an open-source software framework for cloud computing that implements what is commonly referred to as Infrastructure as a Service (IaaS). IaaS are systems that give users the ability to run and control entire virtual machine instances deployed across a variety of physical resources [91]. Using Eucalyptus adds to the openness of Algorizmi and enables the research community to inspect not only the code of Algorizmi but also that of Eucalyptus. Therefore, public scrutiny is achievable. Eucalyptus uses computational and storage infrastructure commonly available to academic research groups to provide a platform that is modular and open to experimental instrumentation and study. Algorizmi's Eucalyptus cloud has multiple physical nodes capable of hosting various virtual machine instances created by the users for their experiments. The goal is to depend on commodity hardware to reduce the cost of implementation and maintenance. The more nodes added to Algorizmi's Eucalyptus cloud, the more experiments Algorizmi can host. Due to all the previous reasons, Eucalyptus is the most suitable choice that will serve the purpose of Algorizmi the best.

In addition to serving as a cloud infrastructure, Algorizmi's Eucalyptus cloud will store user accounts information for all Algorizmi users. A user will only be able to access and/or edit her own account information. When a user requests a new Algorizmi account, she will be provided with a set of credentials:

1. **X.509 certificate [62]** : used to interact with Algorizmi's Eucalyptus cloud through tools that require X.509 certificates, such as Amazon's EC2 command-line tools [5].
2. **Query ID and Secret key:** used with tools that utilize the query interface of Eucalyptus in which requests and parameters are encoded in the URL.

In addition to user account handling, Algorizmi's Eucalyptus cloud offers a data storage service called *Walrus* that is interface compatible with Amazon Simple Storage Service (S3) [6]. Walrus acts as a storage service for virtual machine images (VMIs) used by various users to create virtual machine instances to build their experiments. Root file system as well as kernel and ramdisk images used to instantiate VMIs on nodes can be uploaded to Walrus and accessed from nodes. Algorizmi offers an initial set of VMIs to start populating the database, then users are able to manage (add/delete/control visibility) VMIs they have access to.



Due to some problems that we faced during the implementation of Algorizmi (see Appendix A) we chose our Eucalyptus-based cloud to be the Eucalyptus Public Cloud (EPC) [7]. However, Algorizmi is compatible with any cloud infrastructure that is compatible with Amazon EC2 [5]. Using the EPC imposed some limitations on Algorizmi’s Eucalyptus-based cloud infrastructure due to EPC’s Service Level Agreement (SLA):

1. VMIs are time-limited to 6 hours (they are terminated after that without any warning).
2. No more than 4 VMIs from one user are allowed at any time.
3. VMIs are assigned public IP addresses, but only incoming network connections and inter-instance network connections are allowed.
4. A user can have a maximum of 5GB of permanent storage on EPC.

Those limitations affected the way users can create large scale experiments on Algorizmi and having different network topologies for their experiments. More effects are discussed in Chapter 6.

## Algorizmi Database

Since Eucalyptus only stores data that is related to the proper functioning of the cloud environment it is responsible for, Algorizmi has to manage a database that stores all other data related to its own functionalities. We host this database on the same machine that hosts the Eucalyptus cloud controller. The Algorizmi database repository maintains the following entities:

1. **Experiments:** responsible for storing any experiment-related configuration, settings and the initial state of the experiment to allow results reproducibility.
2. **Datasets:** stores any generated dataset so that all users of Algorizmi are able to re-use it.
3. **Tools:** responsible for storing executables for a plethora of tools used by the system; e.g, sniffing tools, data trace collection tools, etc.
4. **Attacks:** holds attack scripts used to launch attacks against host machines in an experiment. Algorizmi users are able to add their own attack scripts to this database and make them available for other users to maintain an up-to-date state of attacks and offer different types of attacks to match user needs.

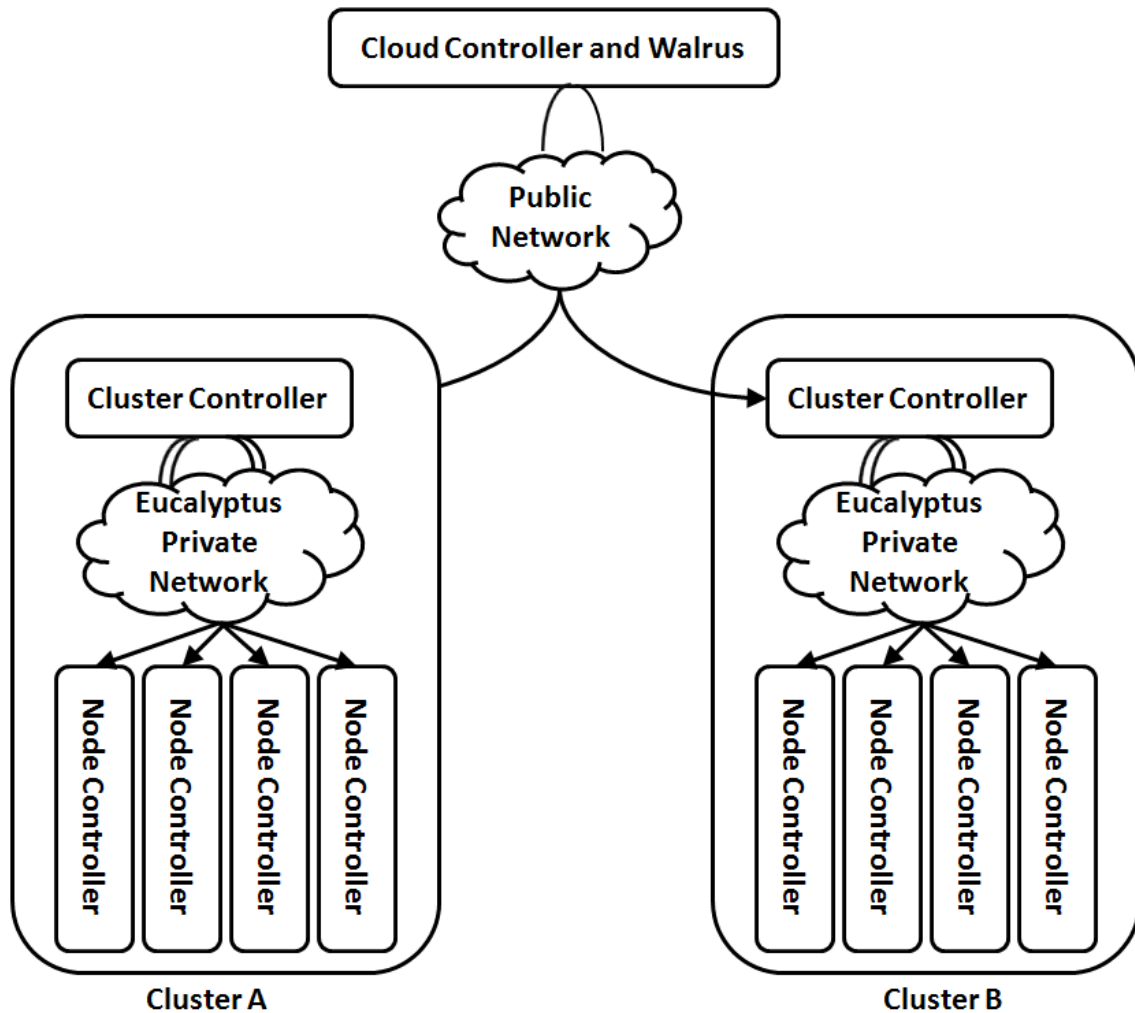


Figure 4.3: A Typical Eucalyptus Installation

#### 4.4.2 Eucalyptus

Algorizmi interacts with its Eucalyptus-based cloud environment through a Eucalyptus installation that manages all the underlying resources and data. A typical Eucalyptus installation allows users to start, control, access, and terminate entire virtual machine instances using an emulation of Amazon EC2s SOAP [4] and Query [3] interfaces. That is, users of Eucalyptus interact with the system using the exact same tools and interfaces that they use to interact with Amazon EC2 [5]. Algorizmi uses the Query interface to interact with its Eucalyptus cloud since this is the option supported by the Java library

for Amazon EC2 [10] used to implement Algorizmi. Figure 4.3 shows that the hierarchy of a typical Eucalyptus installation is comprised of four main components:

1. **Node Controller:** controls the execution, inspection, and termination of VMIs on the host where they run.
2. **Cluster Controller:** gathers information about and schedules VMI execution on specific node controllers, as well as manages virtual instance network.
3. **Storage Controller (Walrus):** is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.
4. **Cloud Controller:** is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes high-level scheduling decisions, and implements them by making requests to cluster controllers.

#### 4.4.3 Algorizmi Manager

This is a major component in Algorizmi since it is the part of the system that orchestrates the work of other components and manages the physical resources made available by Algorizmi's Eucalyptus cloud. Algorizmi Manager will be responsible for the following tasks:

1. Instantiate an Experiment Control for each experiment upon its launch.
2. Receive various configuration data (e.g., attack traffic configuration) and respond to them accordingly.
3. Ensure the correct application of user access rights to data.
4. Terminate any out-of-control experiment.
5. Offer a proper Application Programming Interface (API) to interact with a Eucalyptus cloud. This facilitates the process of building a GUI for Algorizmi as an independent application that is built on top of this API. Such a feature allows other members of the research community to build other versions for this GUI if desired.
6. Interface with Metasploit to generate attack traffic for an experiment. More details about generating attack traffic from MSF is given in the following subsection.

#### 4.4.4 Application Components

This layer includes all application components running on top of the Algorizmi Manager. It has several functionalities including:

#### Resources Monitoring

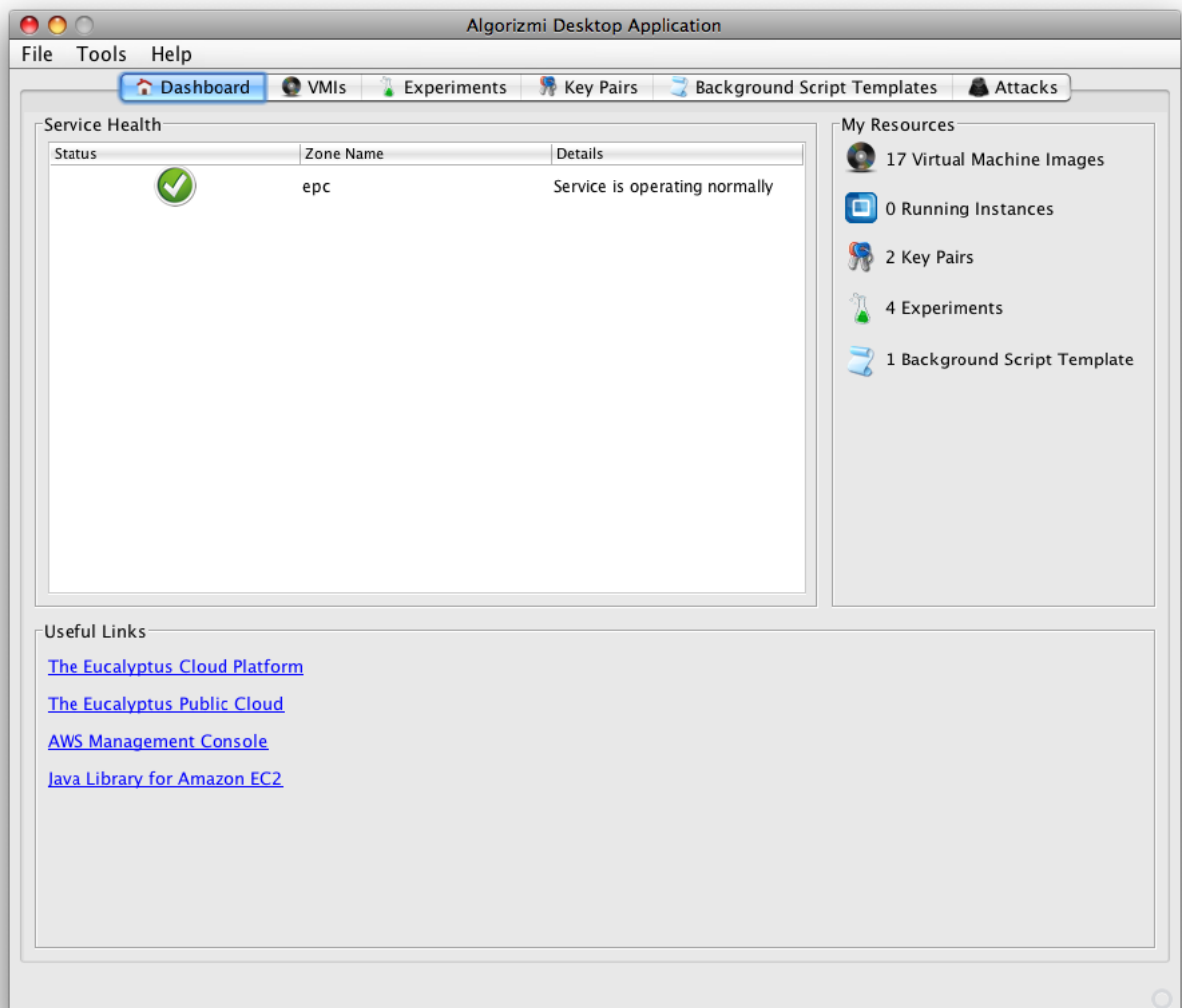


Figure 4.4: A snapshot of Algorizmi Dashboard Tab

This component is responsible for providing each Algorizmi user with an up-to-date report of the resources available to her. The resources include the following:

1. Available clusters in Algorizmi's Eucalyptus cloud.
2. Virtual machine images the user can access and that can be used to instantiate VMIs for a given experiment.
3. VMIs currently run by the user.
4. SSH key pairs owned by the user.
5. Experiments created by the user.
6. Background script templates available in Algorizmi to help generate background traffic.

Figure 4.4 is a snapshot of the Dashboard Tab of Algorizmi Desktop Application. The Algorizmi Dashboard consists of three panels:

1. **Service Health Panel:** shows the status of all clusters in the Eucalyptus-based cloud infrastructure of Algorizmi and details about each cluster.
2. **My Resources Panel:** reports to the user the resources she currently has control on, and the current resources available through Algorizmi.
3. **Useful Links Panel:** provides Algorizmi users with a set of useful links in a case a user wants to get more details about tools used to develop Algorizmi.

## SSH Key Pairs Management

Algorizmi users are provided with tools to create SSH key pairs in case they would like to have access to the VMIs running within their experiments from outside Algorizmi. Each user can create a new SSH key pair or delete any of her existing SSH key pairs.

Figure 4.5 is a snapshot of the Key Pairs Management Tab of the Algorizmi Desktop Application. The snapshot shows that the Key Pairs Management Tab consists of one table that shows all the keypairs of the current user. Each key pair is represented as a row in the table, with one column displaying the SSH key pair name and the other column displaying the fingerprint of the SSH key pair. In addition, there is a toolbar that allows the user to create a new SSH key pair or delete the selected SSH key pairs from the table, and consequently from the system.

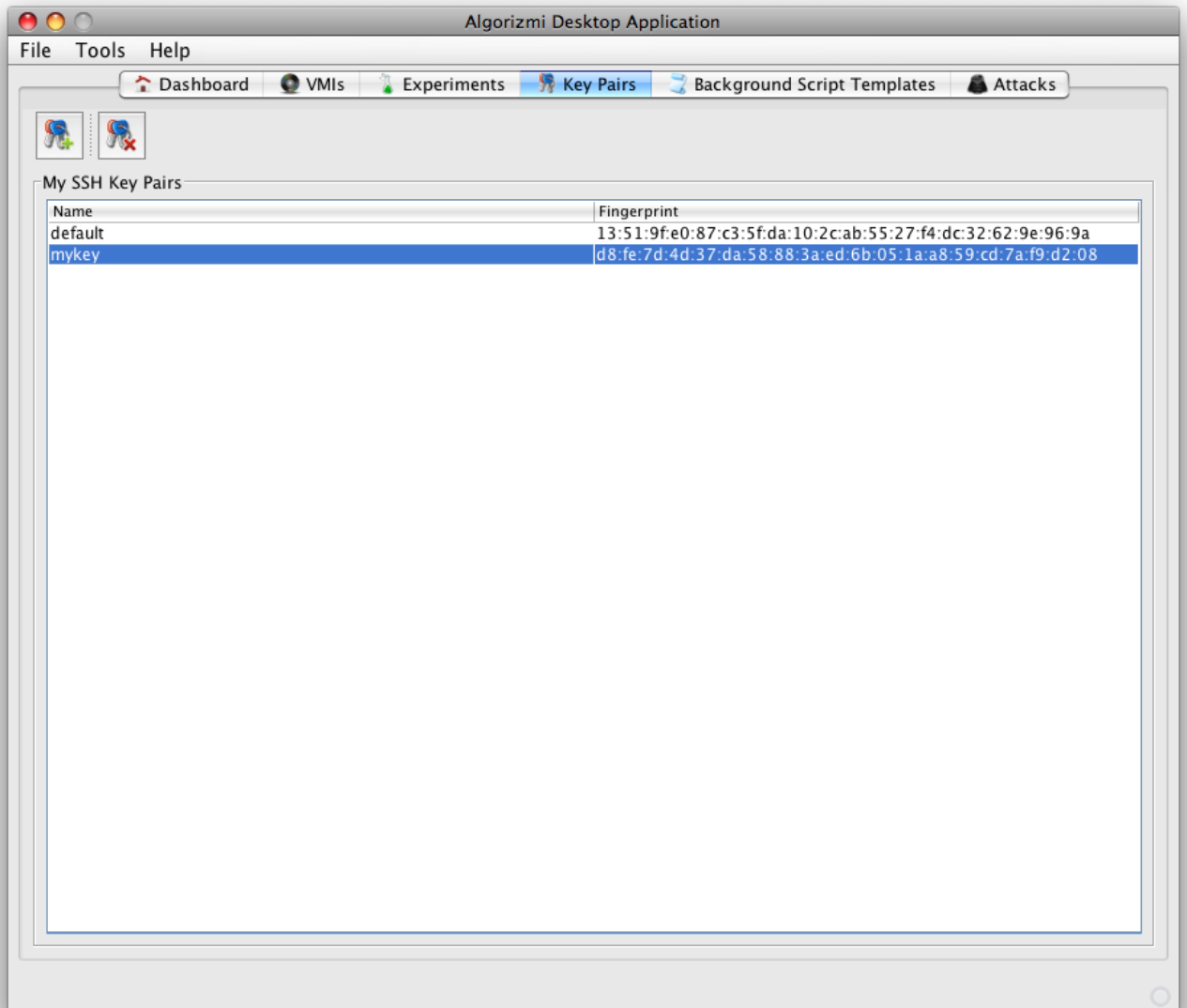


Figure 4.5: A snapshot of Algorizmi Key Pairs Management Tab

## Virtual Machine Images Management

Virtual machine images play an important role in Algorizmi since they are the means of instantiating VMIs to build an Algorizmi experiment. A virtual machine image is a software implementation of a machine (i.e. a computer) that executes programs like a

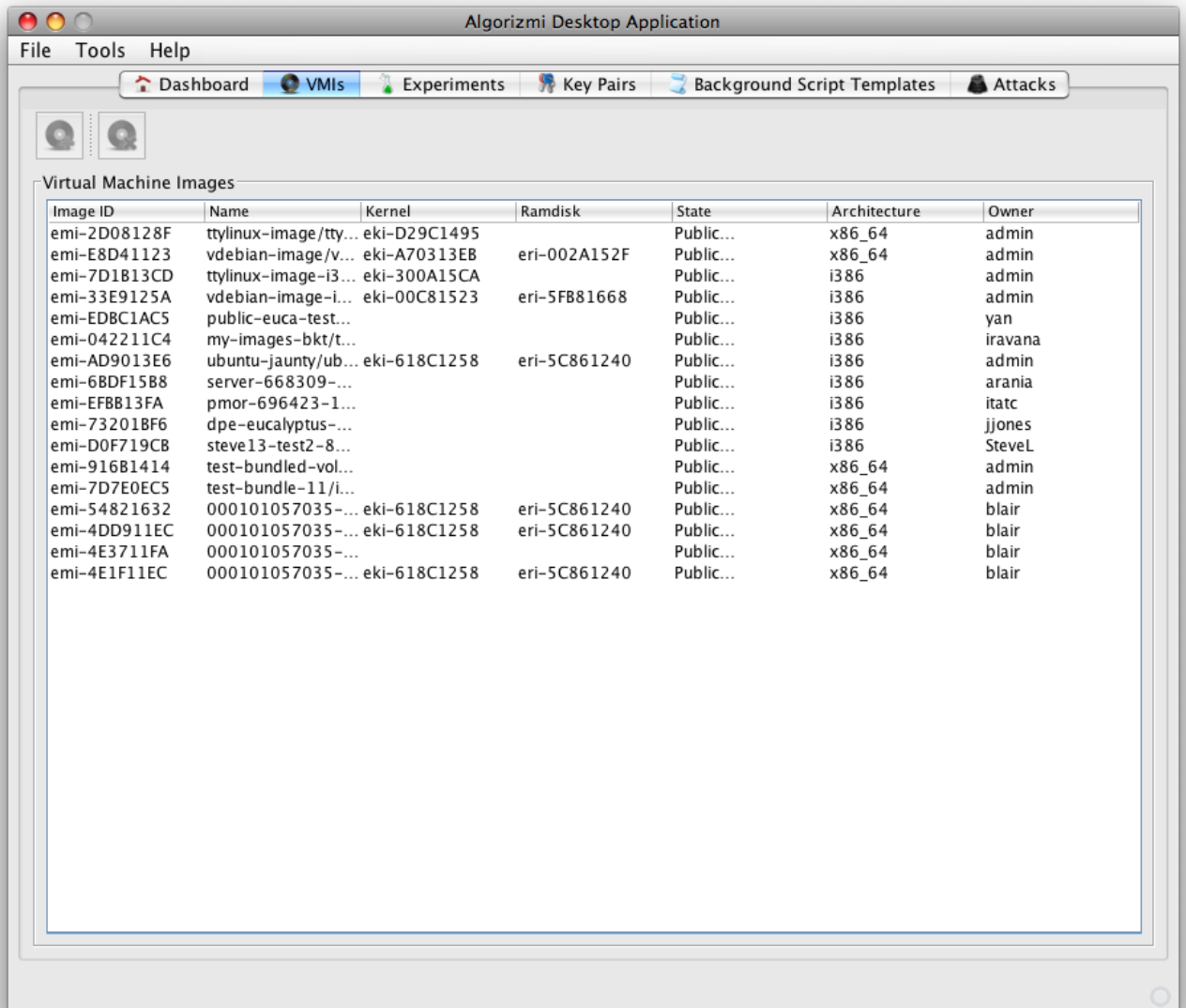


Figure 4.6: A snapshot of Algorizmi Virtual Machine Images Management Tab

physical machine. Algorizmi provides its users with an initial set of virtual machine images to help them build basic experiments. If this initial set is not sufficient, a user can bundle and upload her virtual machine image (with her choice of kernel and ramdisk) to Algorizmi. She can also make it available to other users to benefit from if she desires. A user can also unbundle (delete) any of her existing virtual machine images.

Figure 4.6 shows the Virtual Machine Images Tab of the Alorizmi Desktop Application. The Virtual Machine Images tab consists of a table that lists all the available virtual machine images in Alorizmi. Each row provides the following details of a virtual machine image:

1. **Image ID:** a unique identification for the virtual machine image.
2. **Name:** the name given to the virtual machine image by its owner.
3. **Kernel:** the kernel image associated with the virtual machine image.
4. **Ramdisk:** the ramdisk image associated with the virtual machine image.
5. **State:** the visibility of the virtual machine image (public/private).
6. **Architecture:** the system architecture that the virtual machine image is compatible with.
7. **Owner:** the owner of the virtual machine image.

The Virtual Machine Images tab also has a toolbar to allow Alorizmi users to add/remove virtual machine images. However, the toolbar is disabled in the current version of Alorizmi Desktop Application due to some limitations in the EPC. Despite not being able to add new virtual machine images to the current installation of Alorizmi, the users can use the images currently offered by the administrators of Eucalyptus.

## Background Traffic Management

In order to help simulate background traffic (i.e normal traffic that contain no attacks), Alorizmi provides its users with background script templates which can be used as initial models for the background scripts they would like to run in their own experiments. A user can create new templates if she wants or delete any of her existing templates as well. Each background script template is, in fact, a shell script giving the users a multitude of options when developing a background script template. In other words, a background script template can be any script that requires an interpreter, e.g. unix shell script [9], php script [16], perl script [15], python script [18], Expect script [8], etc.

Figure 4.7 is a snapshot of the Background Script Templates tab of the Alorizmi Desktop Application. This tab consists of three components:

1. **Background Script Templates Toolbar:** allows the user to add/remove background script templates.



2. **Background Script Templates Panel:** consists of a table that lists the currently available background script templates in Algorizmi. Each row provides the numeric identifier and the name of a background script template.
3. **Selected Background Script Template Panel:** displays the details of the selected background script template including the background script template numeric identifier, its name and the script itself.

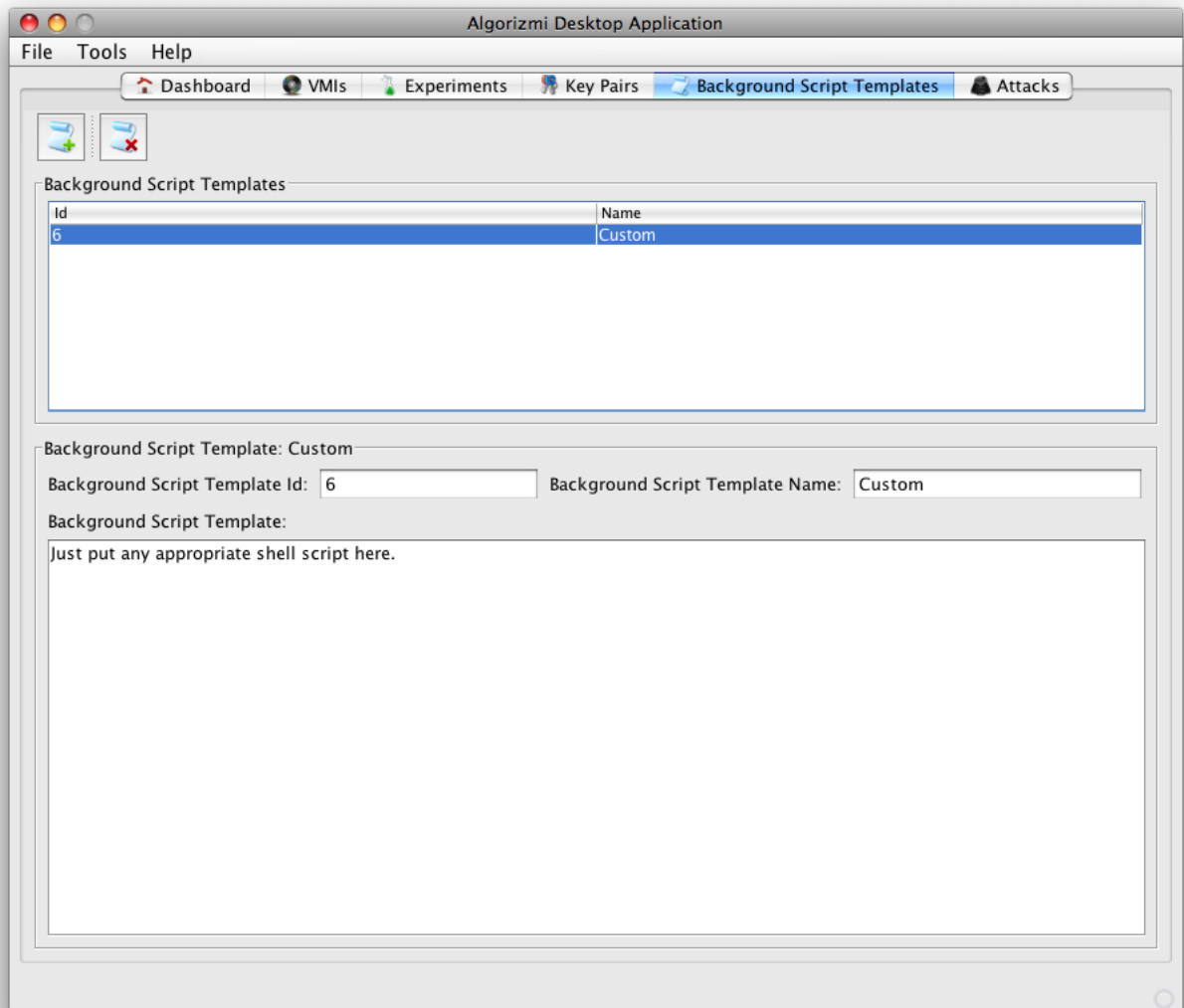


Figure 4.7: A snapshot of Algorizmi Background Script Templates Management Tab

## Attack Traffic Management

In order to properly evaluate an IDS algorithm, it has to be tested against a dataset that contains network attacks. We chose Metasploit [11] to provide Algorizmi users with this functionality. Metasploit is an open-source project that provides useful information to people who perform penetration testing, IDS signature development, and exploit research. The tools and information Metasploit offers are provided for legal security research and testing purposes only. Metasploit is now run by Rapid7 [26], but is still an open-source project. Algorizmi users are able to select and configure the attacks they would like to launch in their experiments from a wide variety of exploits offered by Metasploit that cover almost all network attack types. If a user is not content with the set of exploit modules Metasploit provides, she can write her own Metasploit attack module and use it instead. She can also suggest her newly built exploit module to be added to future releases of Metasploit.

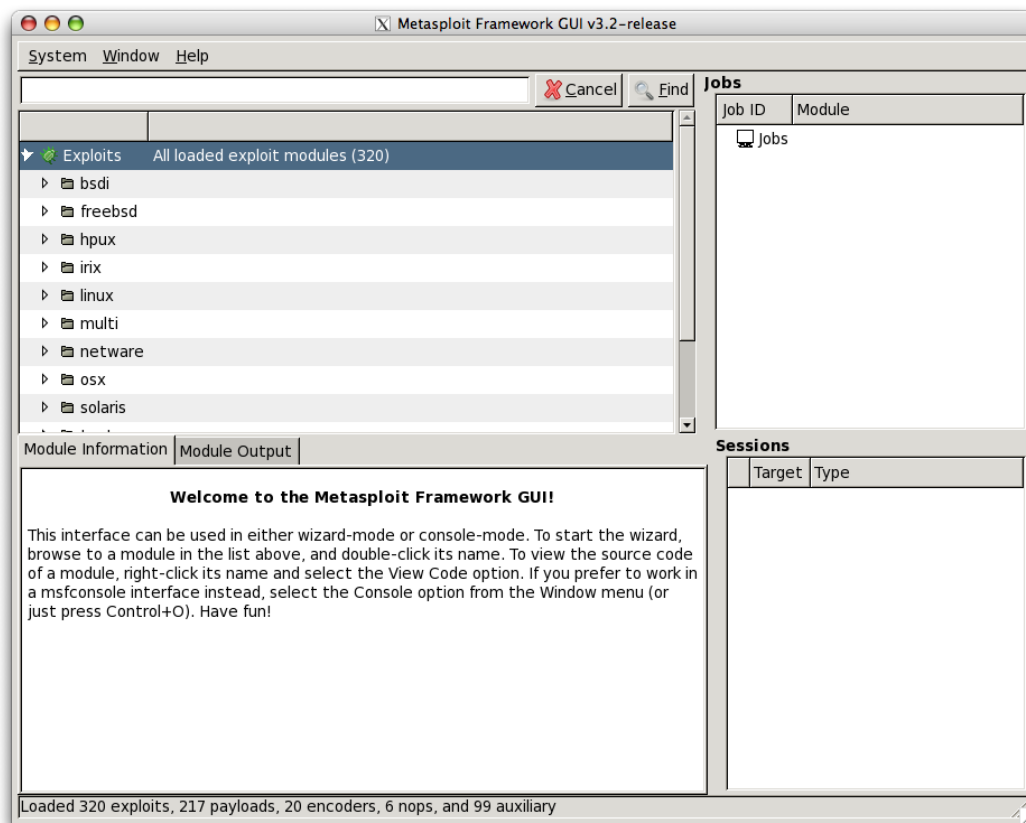


Figure 4.8: A snapshot of Metasploit GUI

Figure 4.8 is a snapshot of the Metasploit Framework desktop GUI. The desktop GUI provides users with the ability to select an exploit module from all the available exploits. A user can then configure this module to satisfy her requirements. The GUI also has an output panel that shows the output of the running attack modules. In addition, a user can monitor her various attack sessions through the **Sessions** panel. Finally, a user can keep track of her attack jobs through the **Jobs** panel.

## Experiment Management

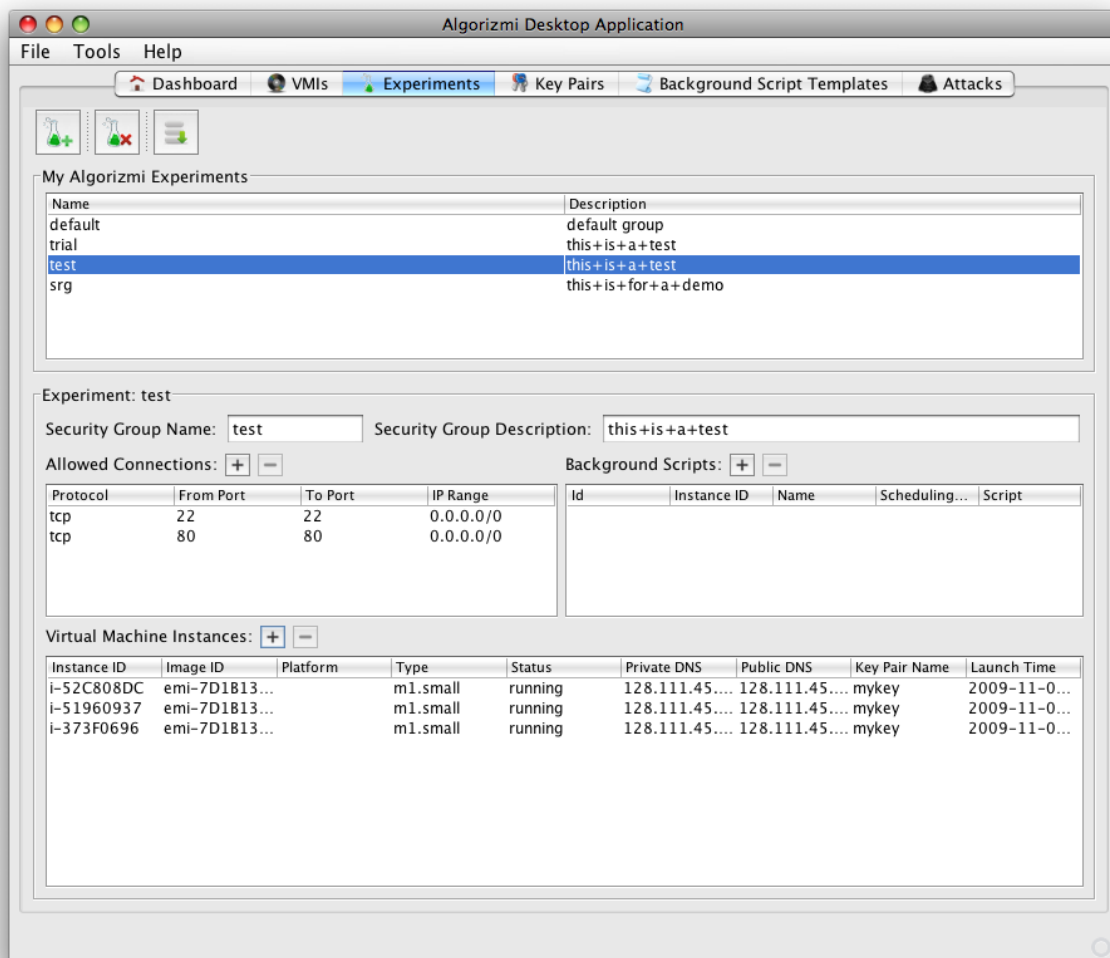


Figure 4.9: A snapshot of Algorizmi Experiments Management Tab

An Algorizmi user can manage her experiments in various ways. She can create a new experiment with a unique name and a short description for the experiment. She can delete any number of her existing experiments at any time she wishes. Figure 4.9 is a snapshot of the Experiments tab of the Algorizmi Desktop Application. The figure shows that this tab consists of three major components:

1. **Experiments Tab Toolbar:** allows the users to create new Algorizmi experiments, delete any of their existing experiments and download the dataset of a given experiment.
2. **My Algorizmi Experiments Panel:** consists of a table that shows all the experiments owned by the current Algorizmi user. Each row in the table represents the details of an experiment, namely the experiment name and description.
3. **Selected Experiment Panel:** provides the user with detailed information of the selected experiment. This includes: the security group the experiment belongs to; allowed connections to/from the experiment with means to grant new rules or revoke existing rules; details of background scripts running in the experiment with means to add new background scripts or remove existing ones; detailed information about the VMIs building up the experiment network with means to instantiate new VMIs or terminate existing ones.

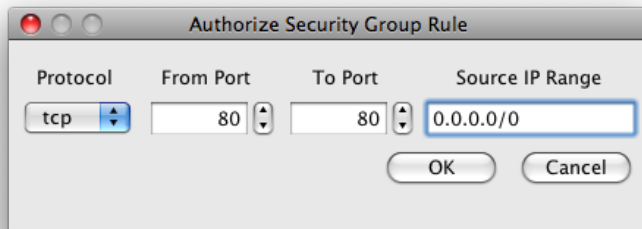


Figure 4.10: Authorizing a new network ingress rule

An Algorizmi user can alter the network ingress rules for an experiment of hers (affecting all VMIs running within that experiment) to authorize/revoke certain network connections. Figure 4.10 shows the dialog that is presented to the user when she requests to authorize a new network ingress rule (i.e. a new security group rule). The user is required to specify the following security group rule attributes:

1. **Protocol:** the protocol that will be subject to this rule (tcp/udp/icmp).
2. **From Port:** the beginning of the port range that this rule will be applied to.
3. **To Port:** the end of the port range that this rule will be applied to.
4. **Source IP Range:** the source from which this type of connection will be allowed. The source can be either a range of IP addresses or the name of another experiment in Algorizmi.

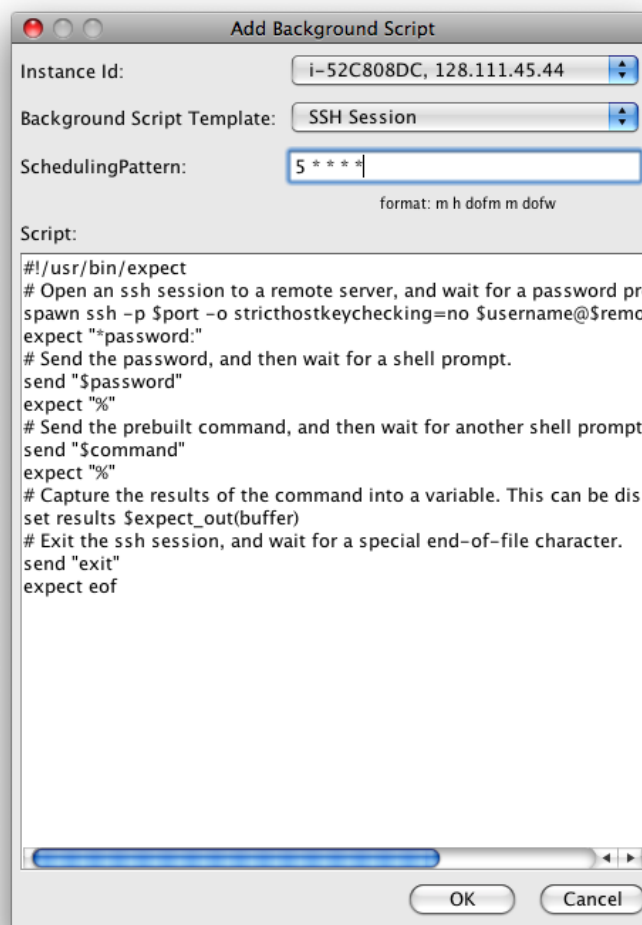


Figure 4.11: Adding a new background script to an experiment

An Algorizmi user can also add/delete background scripts from any of her experiment. If a user wants to add a new background script to run on her experiment, she will be presented with the dialog shown in Figure 4.11 where she has to specify four attributes:

1. **Instance Id:** the identifier of the running VMI in the experiment that will run the background script augmented with the instance IP address.
2. **Background Script Template:** the template this script is based on (to facilitate the writing of the script).
3. **Scheduling Pattern:** the scheduling pattern of the script which will eventually allow Algorizmi to generate the appropriate crontab entry [71] for the script on the selected VMI.
4. **Script:** the background script itself.

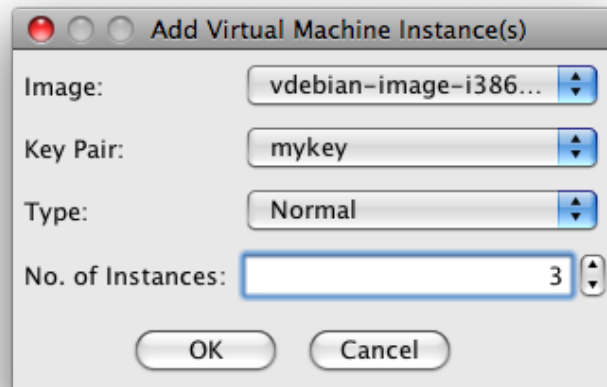


Figure 4.12: Adding a VMI to an experiment

Finally, an Algorizmi user can instantiate/terminate VMIs on any of her experiments given that there are some available resources. In order to instantiate one or more VMIs, a user will be presented with the dialog shown in Figure 4.12 and will have to specify four attributes:

1. **Image:** the virtual machine image she wants to instantiate.
2. **Key Pair:** the SSH key pair she would like to associate with the instantiated VMIs.

3. **Type:** the type of the instantiated VMIs (Normal or Attacking).
4. **No. of Instances:** the number of instances she wants to instantiate of the selected virtual machine image.

The type of the VMI will affect the way Algorizmi configures it. For both types, Algorizmi will install tshark [24] and run it as a system service to gather all incoming/outgoing network traffic. This will eventually contribute to the dataset generated for the experiment. If the selected type is *Normal*, Algorizmi will create the appropriate directories required for running background scripts on the instance and install any required software packages (e.g openssh, expect, php) to allow the instance to function properly. On the other hand, if the selected type is *Attacking*, Algorizmi will create appropriate directories for attack scripts and install the Metasploit Framework.

Finally, a user can generate the dataset of an experiment at any point in time (given that the experiment has some running VMIs). An experiment dataset is a compilation of the network traffic traces gathered from all the VMIs within the experiment. The dataset can be downloaded by the user in more than 20 different output formats (basically all the formats supported by Wireshark [27]). Figure 4.13 shows the dialog that will be presented to the user wishing to generate the dataset of an experiment. In order to generate (and download) the dataset the user should specify the output format of the dataset.

## Graphical User Interface

The GUI is the one component in Algorizmi that has direct interaction with the users. To make it as user friendly, we built a portable GUI for Algorizmi as a Java desktop application. Java allows the GUI of Algorizmi to be executed on various platforms without the need to recompile its code into a different binary for each platform. This increases the spectrum of potential users in the research community. The GUI consists of the following major components:

1. **Menu Bar:** provides the user with access to user settings, help dialog and can be expanded to include more features in the future (e.g quick links to create a new experiment, a new SSH key pair, etc).
2. **Main Panel:** is a tabbed panel where each tab is specified for managing a major component in Algorizmi: VMIs, experiments, key pairs, background script templates and attacks.
3. **Progress Bar:** adds to the user friendliness of the Algorizmi Desktop Application since this bar allows the user to keep track of the progress of the task(s) currently

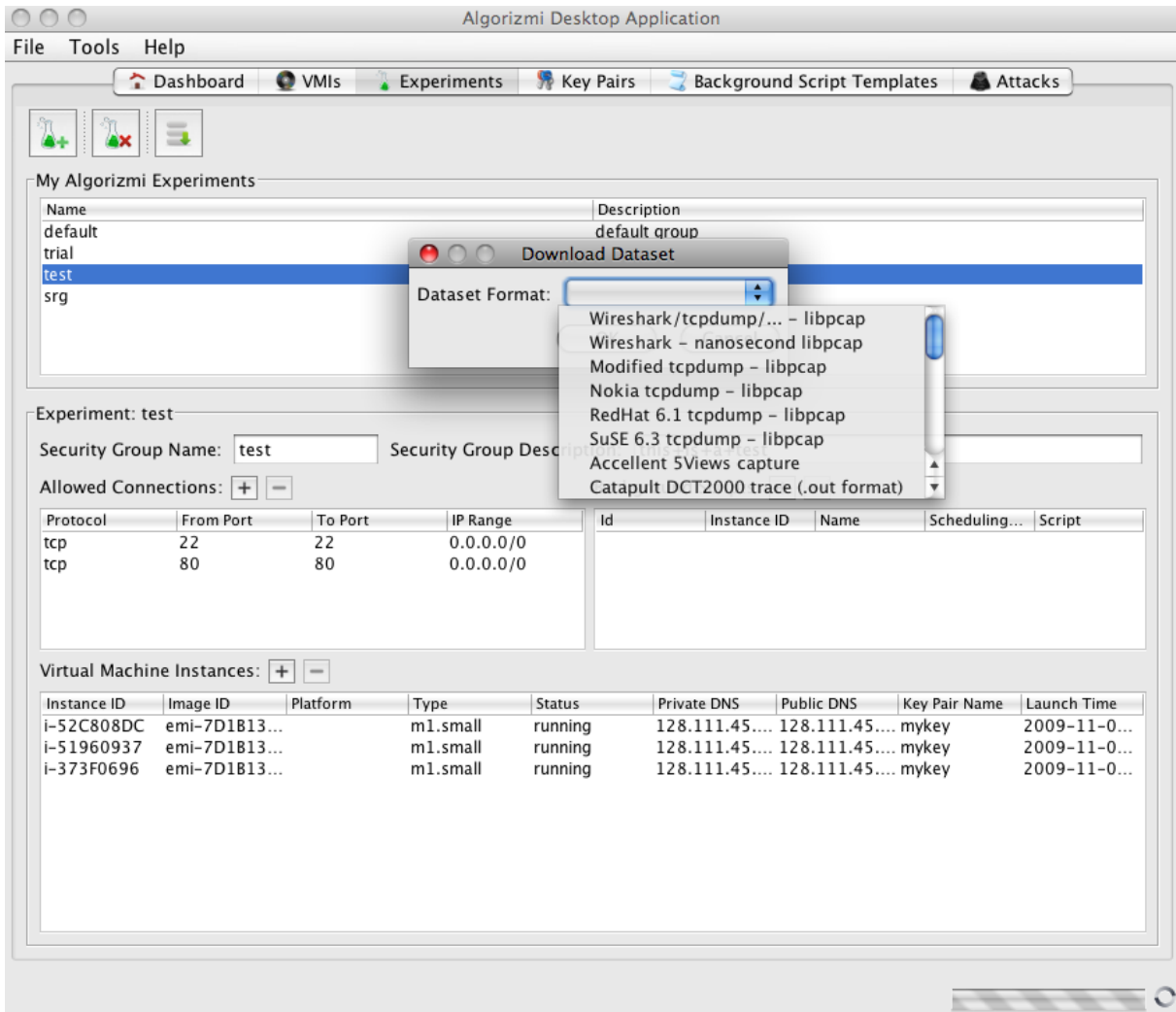


Figure 4.13: Downloading the dataset of an experiment

running. This allows the user to perform more than one task at the same time without one task blocking the application from responding to other user requests.

All figures shown in this section are snapshots taken while running Algorizmi Desktop Application, the GUI we built for Algorizmi.



## 4.5 Using Algorizmi to Generate a Dataset

This section illustrates how a researcher can use Algorizmi in order to generate a dataset that can be used to perform an offline evaluation for an IDS.

### 4.5.1 Create New User Account

Figure 4.14 represents **Step 1** where a user creates a new user account. First, a user should go to Algorizmi's Eucalyptus-based cloud front-end website and apply for a new user account. The user then has to fill out the application form with her username, password, full name and email address. If the user wishes, she can fill out some other optional fields: telephone number, project leader, affiliation and project description. Once the user submits her application form, she will be sent an account activation email within a couple of days to start using her account.

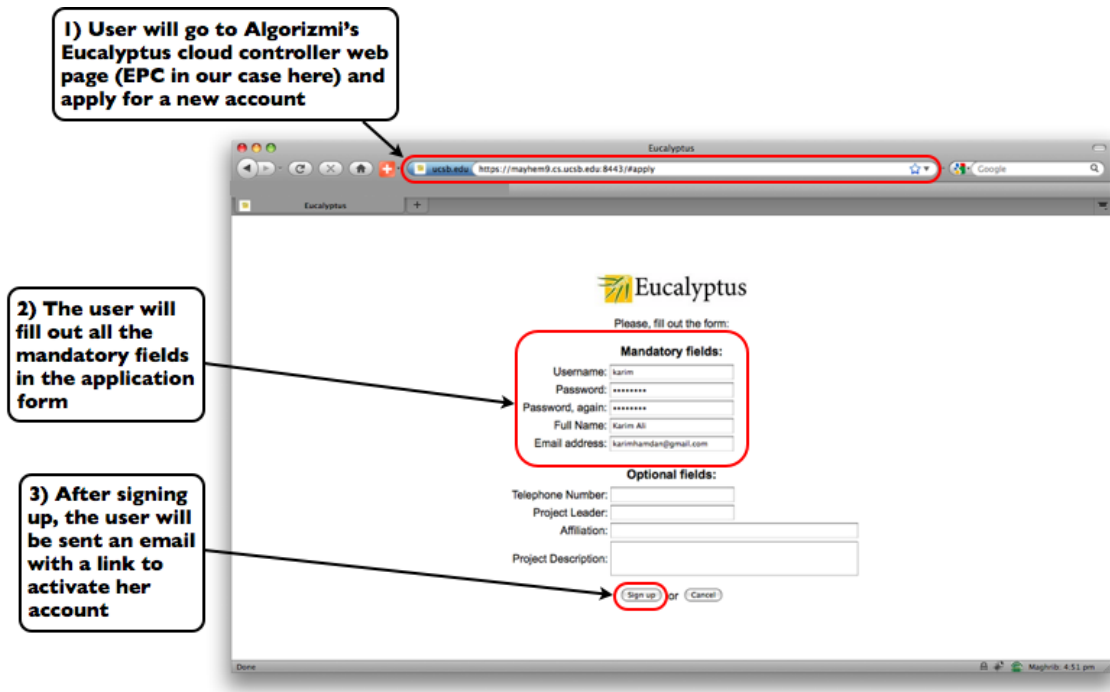


Figure 4.14: Create a new user account

## 4.5.2 Modify User Settings

Figure 4.15 represents **Step 2** where a user modifies her user settings in the Algorizmi Desktop Application. After the user creates her account, she should keep a note of her Query ID and Access Key. She can then plug those values in their corresponding fields in the Algorizmi settings dialog in the Algorizmi Desktop Application. Finally, she applies her changes to be able to start using Algorizmi.

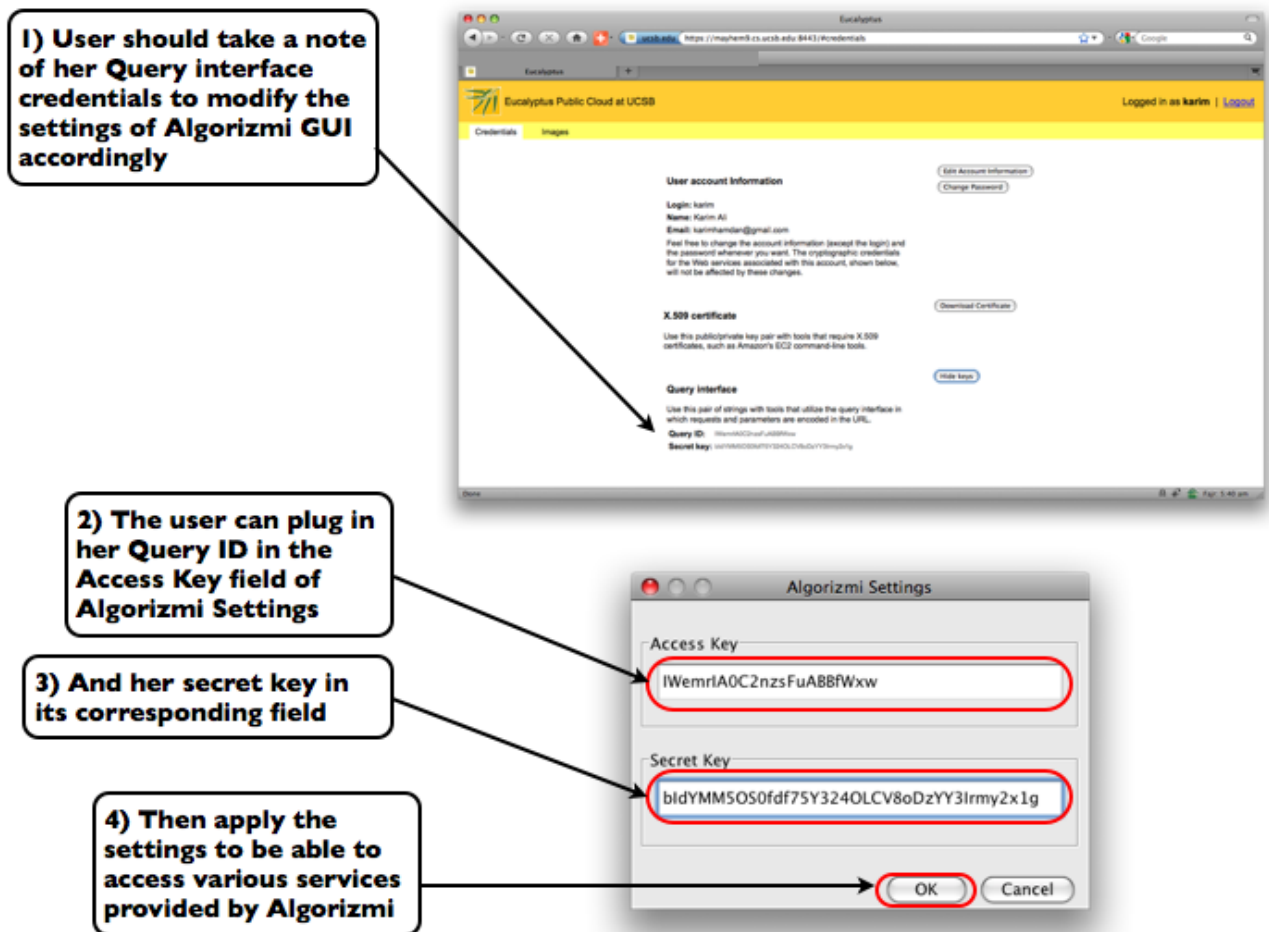


Figure 4.15: Modify user settings

### 4.5.3 Create New Experiment

Figure 4.16 represents **Step 3** where a user creates a new experiment. First, the user selects the *Experiments* tab in the Algorizmi Desktop Application. The user will then find a button in the tab's toolbar that allows her to create a new experiment. Afterwards, the user will be presented with a dialog allowing her to set the name and the description of the new experiment she wants to create. Finally, the new experiment will be created once the user clicks on the *OK* button.

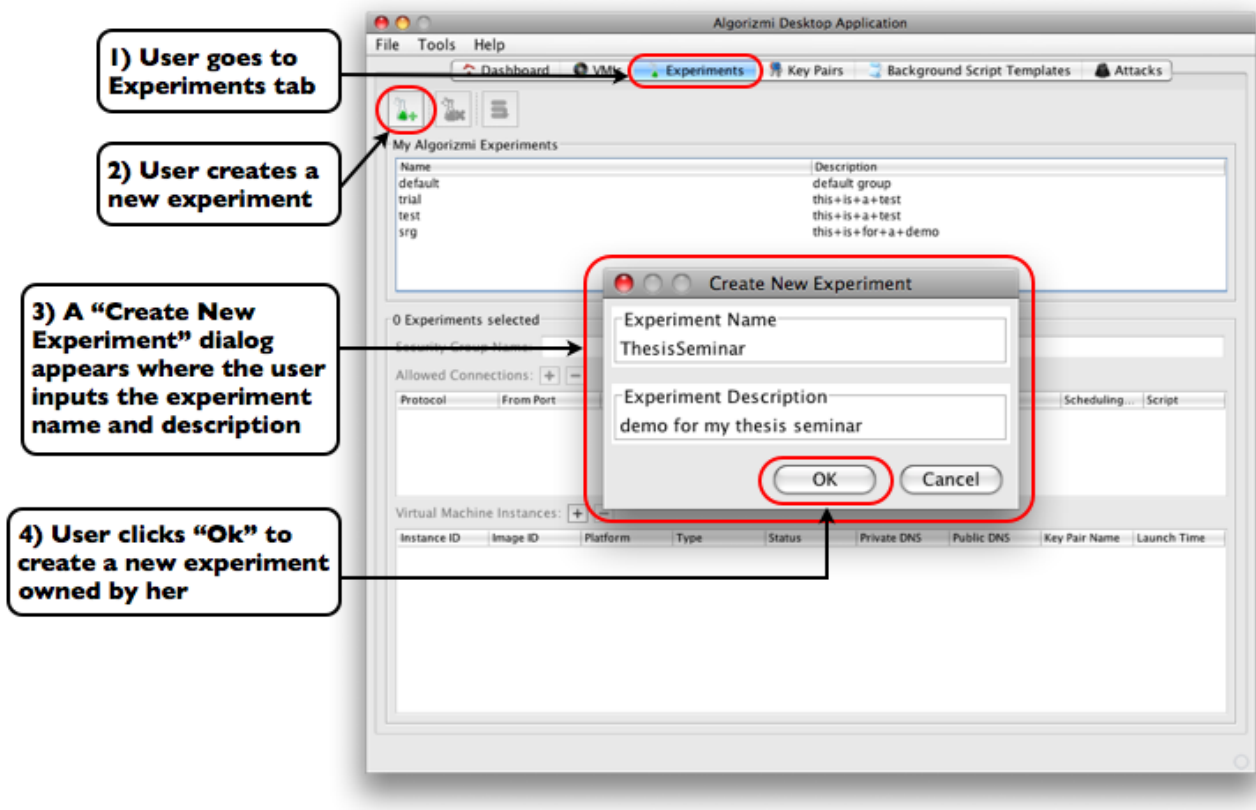


Figure 4.16: Create a new experiment

#### 4.5.4 Add a VMI to an Experiment

Figure 4.17 represents **Step 4** where a user adds a VMI to one of her existing experiments. First, the user selects one of her existing experiments. She then finds an *Add* button above the *VMIs* table in the *Selected Experiment Panel* that when clicked will present her with a dialog box. In the latter, she can specify the image she wants to instantiate, key pair associated with the VMIs, the type of the VMIs and how many VMIs she wants to instantiate. Finally, once the user clicks the *OK* button, the VMIs will be instantiated, configured, and ready for use.

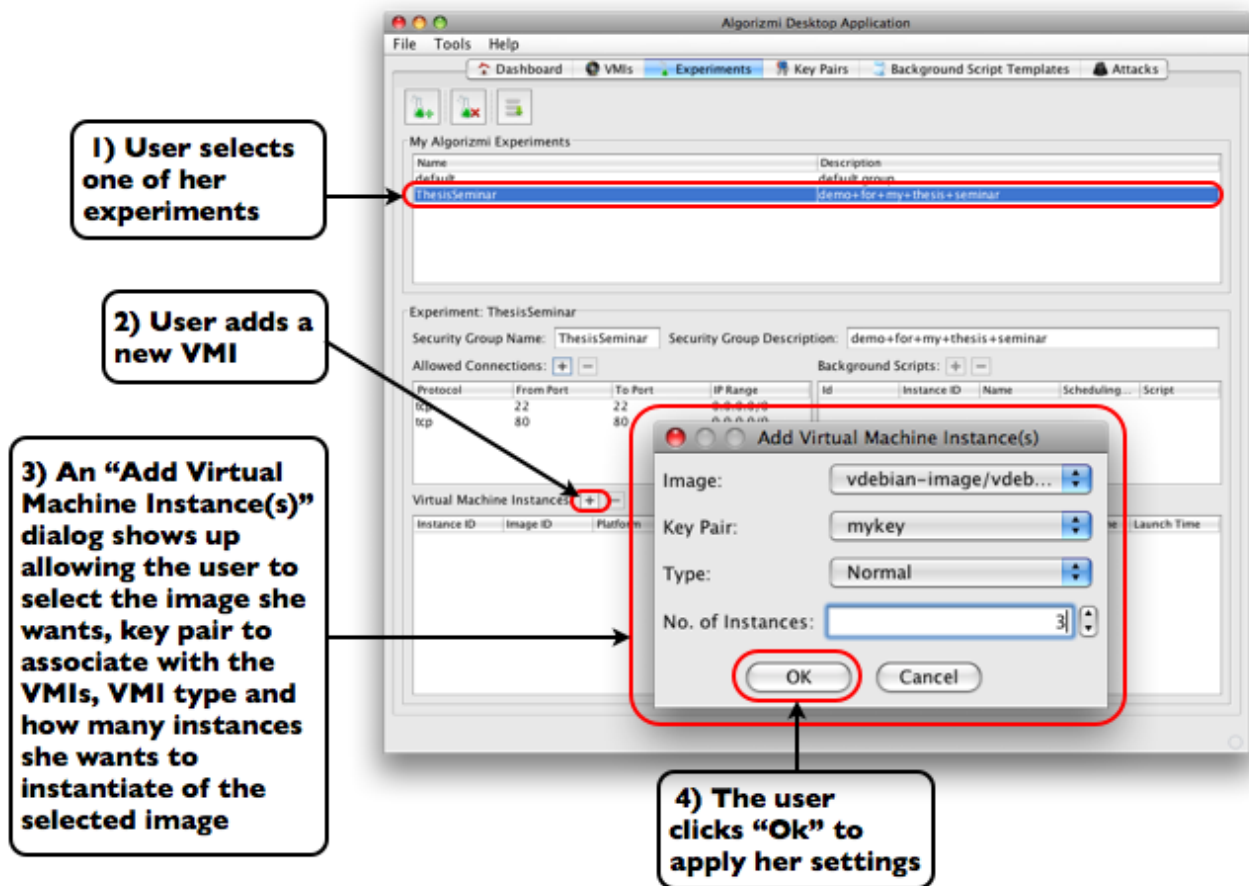


Figure 4.17: Add a VMI to an existing experiment

## 4.5.5 Add a Background Script to an Experiment

Figure 4.18 represents **Step 5** where a user adds a background script to one of her existing experiments. First, the user selects one of her existing experiments. She then finds an *Add* button above the *Background Scripts* table in the *Selected Experiment Panel* that when clicked will present her with a dialog box. In the latter, she can specify the instance on which this background script will be installed, the template this background script is based on, the scheduling pattern of the background script and the script itself. Finally, once the user clicks on the *OK* button, the background script will be created and installed on the VMI the user selected.

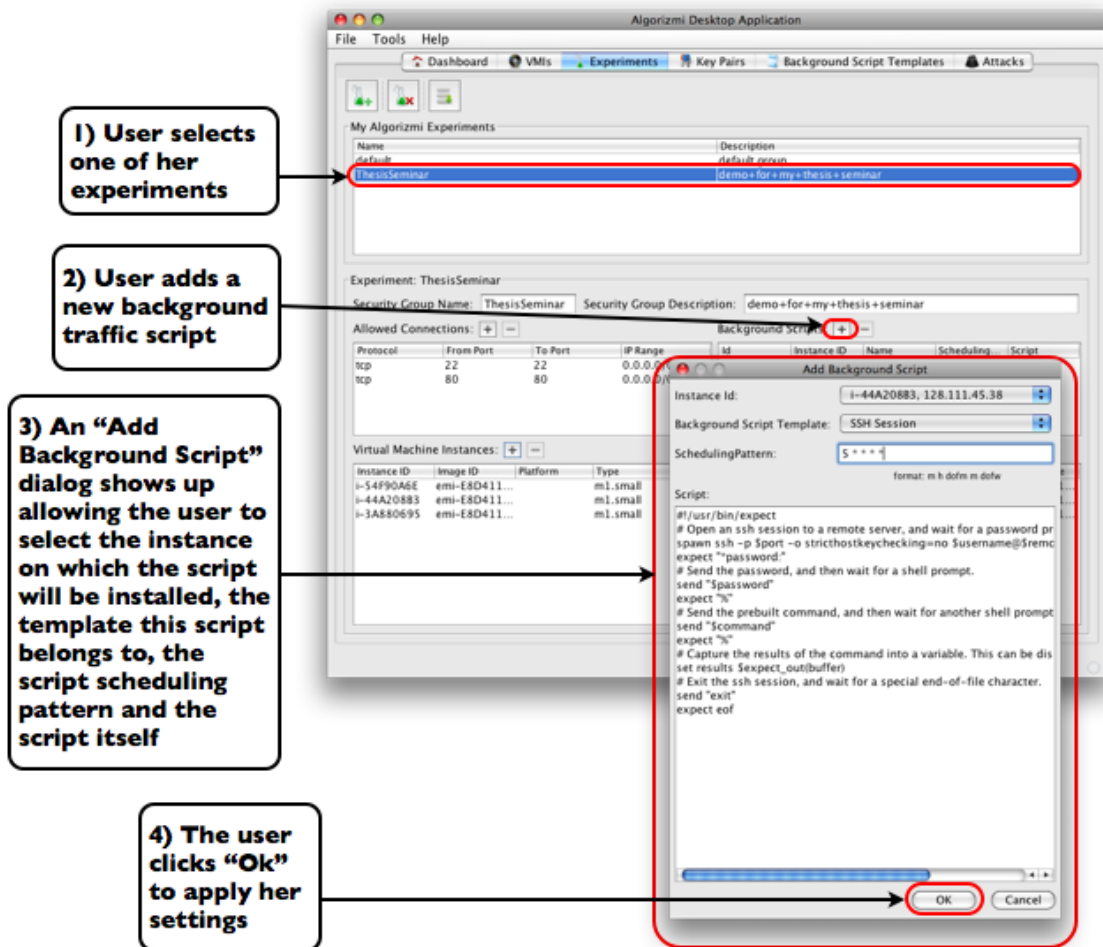
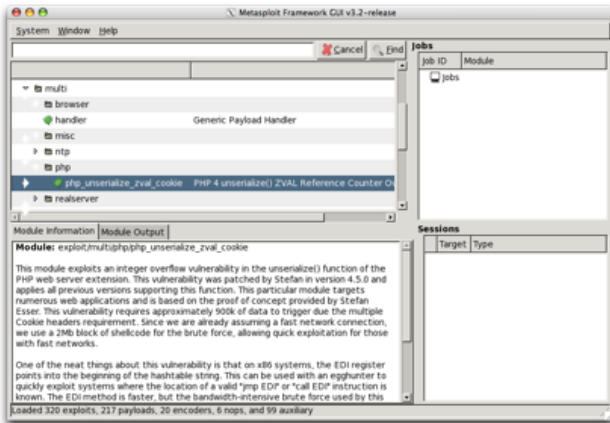


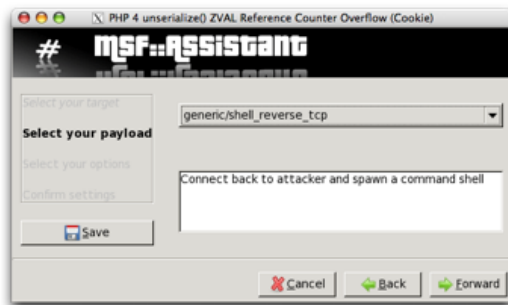
Figure 4.18: Add a Background Script to an existing experiment

### 4.5.6 Generate Attack Traffic

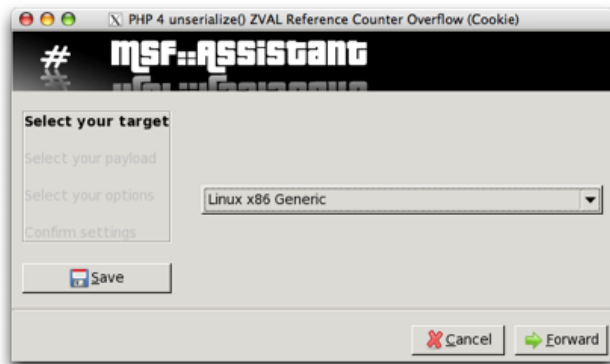
Figure 4.19 represents **Step 6** where a user configures Metasploit to launch attacks against her experiment. First, the user selects one of the attack modules in Metasploit. She then will be presented with a series of configuration parameters that she needs to modify to fit her desired settings. Figure 4.20 shows that once the user is done configuring the attack module, she will be presented with a summary of the attack module configuration with the ability to save that configuration for later use.



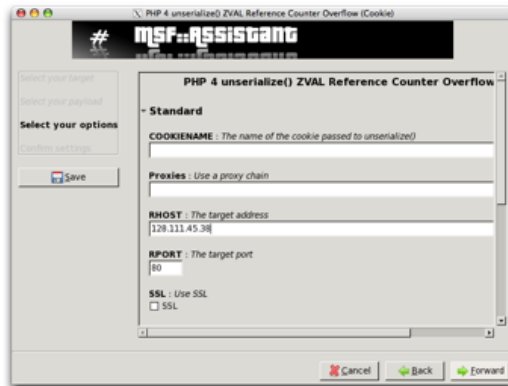
(a) Select attack module



(c) Select payload to be executed when if attack succeeds



(b) Select target of attack



(d) Configure the attack to fit user needs

Figure 4.19: Generate attack traffic from Metasploit

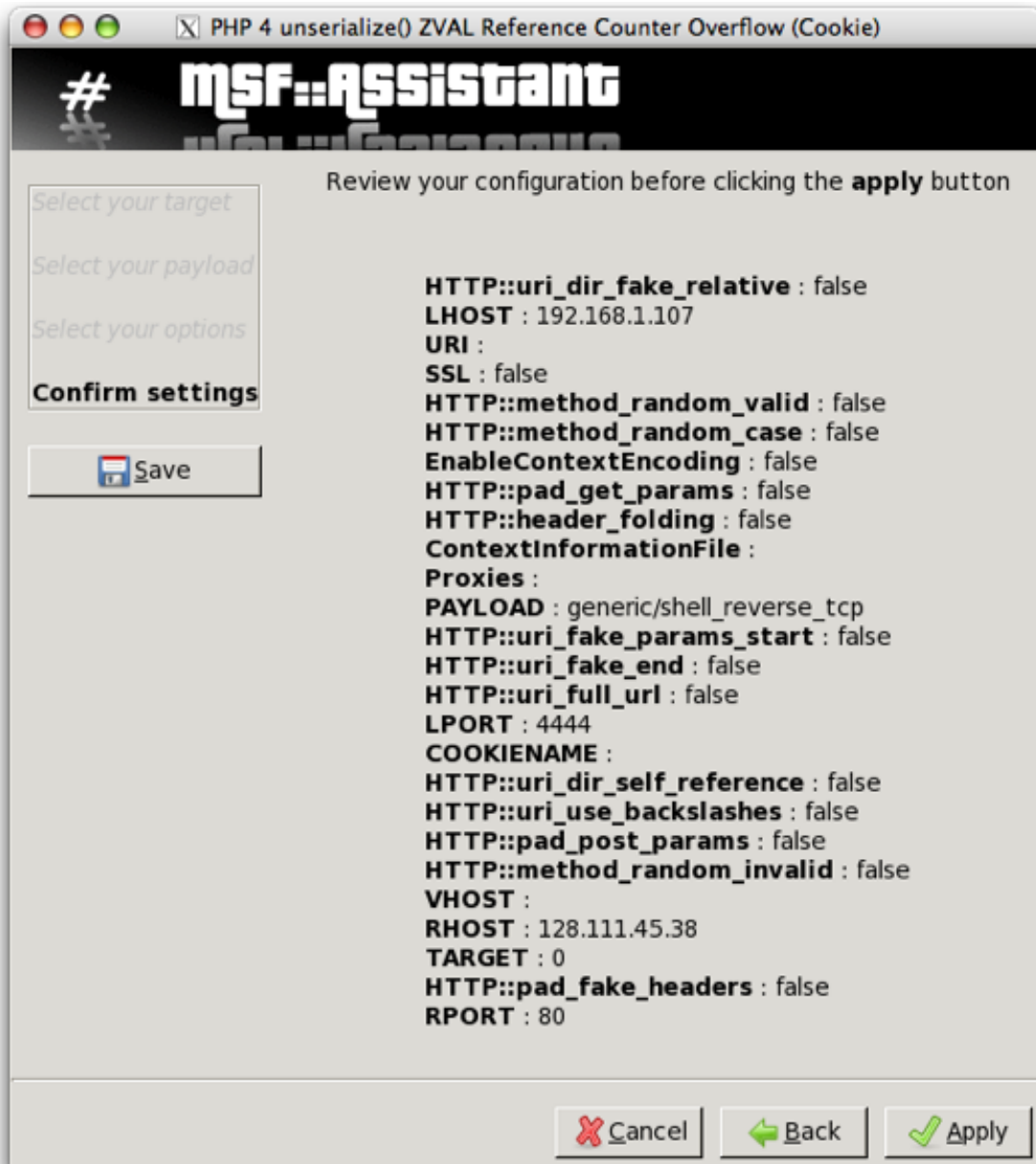


Figure 4.20: Metasploit attack summary

### 4.5.7 Download Dataset

Once a user is satisfied with the current configuration of her experiment and for the period of time network traffic (both background and attack) has been generated, she can ask Algorizmi to generate the dataset of her experiment. Figure 4.21 represents **Step 7** where the user can select the output format of her generated dataset, then the dataset will be downloaded to her home directory. The user can then use the generated dataset to perform an offline evaluation of her IDS.

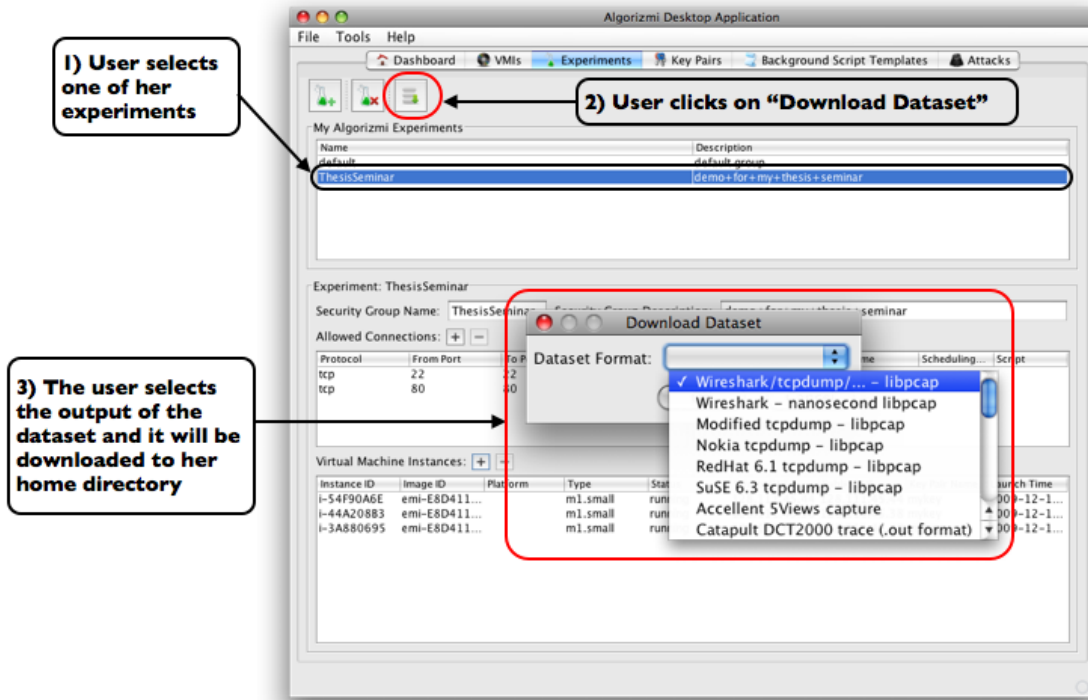


Figure 4.21: Generate and download the dataset for an Algorizmi experiment

## 4.6 Summary

In this chapter, we showed that the architecture of Algorizmi is composed of five layers: infrastructure layer, Algorizmi control layer, experiment control layer, experiment layer, tools layer and the graphical user interface that provides Algorizmi users with a convenient way to interact with Algorizmi. We explained the responsibilities of each layer and its



software components. Finally, we presented a scenario where a user creates an experiment on Algorizmi in order to generate a dataset than can be then used for offline evaluation of an IDS.

# Chapter 5

## Evaluation

### 5.1 Introduction

Our evaluation process involves two parts. First, we validate one of the datasets generated from an experiment running on Algorizmi by running it against Snort and analyzing the results. We then compare Algorizmi to previous research works based on the compliance of each project to the design requirements presented in Chapter 3. In this perspective, we compare Algorizmi to the most recent projects, in addition to the DARPA dataset and using anonymized datasets. Finally, we assess the performance of Algorizmi when handling the common tasks requested by a typical user of the system.

### 5.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 5.3 validates the dataset generation process from one of the experiments build using Algorizmi. Section 5.4 compares Algorizmi with previous research projects based on the research needs each projects satisfies. We assess the performance of Algorizmi in Section 5.5. We conclude the chapter in Section 5.6.

### 5.3 Dataset Validation

In this section, we validate the structure of the generated datasets. We do this by selecting one of our experiments built using Algorizmi and taking its dataset and running it against Snort [106] and checking whether Snort will be able to detect the attacks generated in the

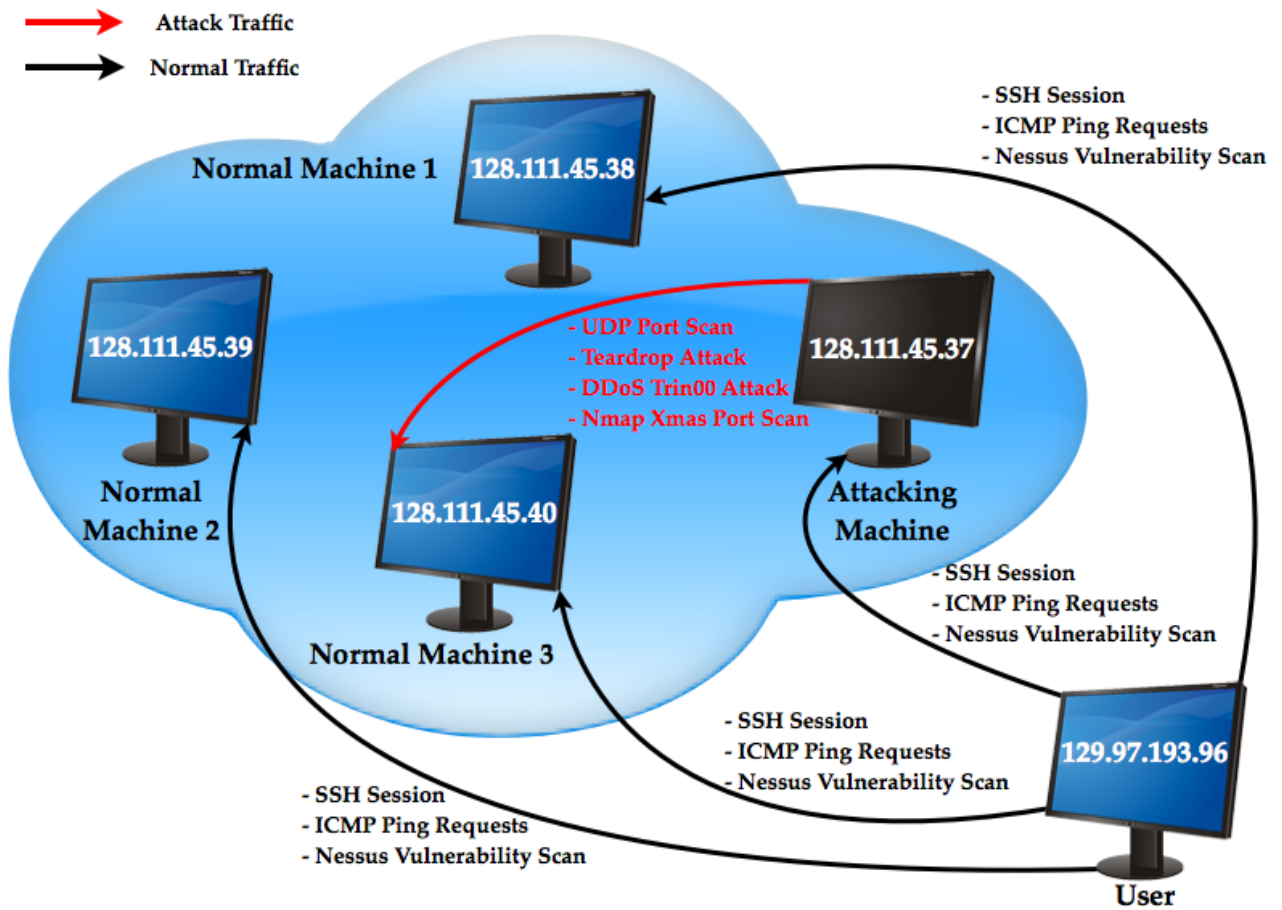


Figure 5.1: Sample Algorithmi Experiment

dataset or not. If the traffic was not generated correctly, Snort will not be able to parse the network traffic traces in the dataset.

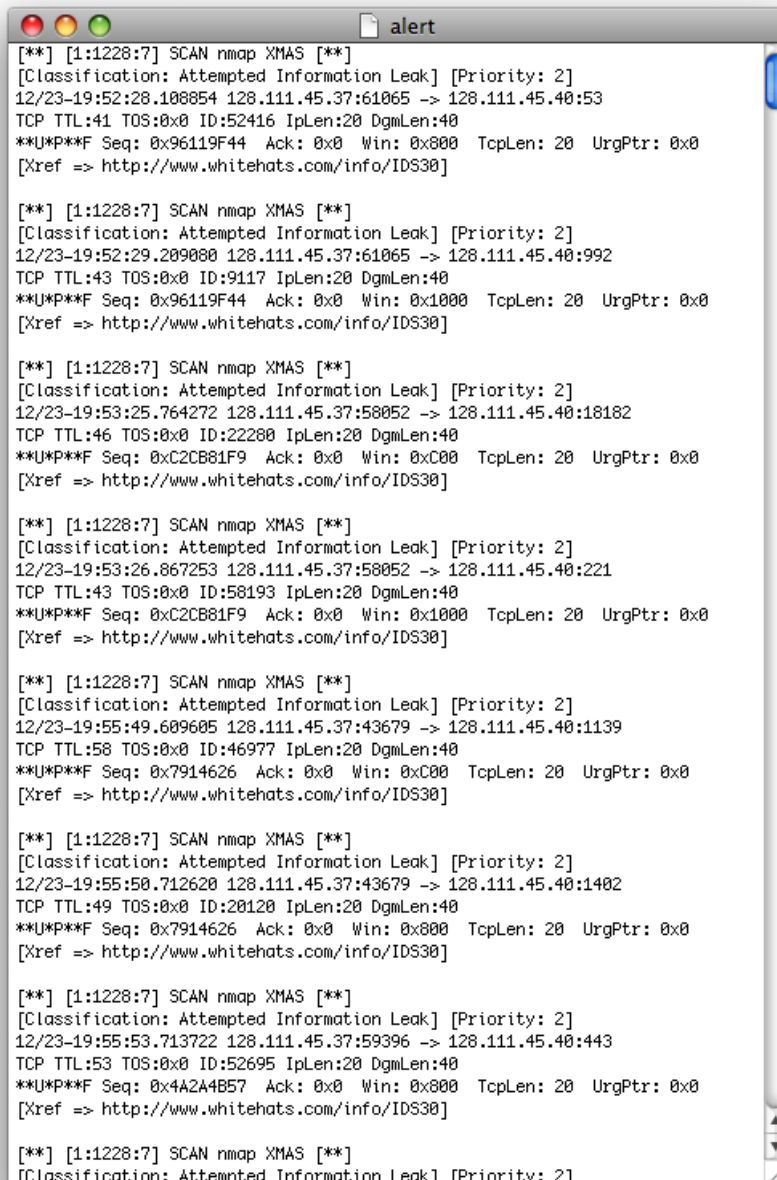
The experiment consists of four VMs, each has Debian 4.0 (code-named *Etch*) installed as the operating system. The VMs are fully connected in a flat network topology as Figure 5.1 shows. All VMs have OpenSSH installed on them to allow the user to access the VMs from outside Algorithmi to perform further configuration if needed. Algorithmi installs Tshark [24] (by default) as a packet sniffer on all of the four VMs so that it will be the tool responsible for capturing the traffic trace on each VMI. Three of the VMs were configured as **Normal** machines (i.e. machines that do not generate attacks) and only one machine was configured to act as an **Attacking** machine. After the VMs were

instantiated and configured to be used, we were able to configure and run some background traffic. We emulated a user who is trying to access the machines within the network and use the services offered by those machines. We sent 10 ping requests from the user machine to each of the VMIs in our network. We established 2 SSH sessions with each of the VMIs using the SSH key pair associated with each VMI when it was first created. Each VMI was scanned using Nessus [12] which also generated other background traffic. In addition, Nessus reported the vulnerabilities found on each VMI. The full Nessus vulnerability report for the four VMIs is available for download [13].

We configured the attacking machine in the network to target its attacks against **Normal Machine 3** with four types of network attacks:

1. **UDP Port Scan:** is a type of scan that probes the target machine looking for open UDP ports [25]
2. **Teardrop Attack:** involves sending mangled IP fragments with overlapping, oversized payloads to the target machine. This can crash various operating systems due to a bug in their TCP/IP fragmentation re-assembly code [22]
3. **Trin00 Attack:** is a set of computer programs to conduct a Distributed Denial of Service (DDoS) attack [23]
4. **Nmap Xmas Port Scan:** sends malformed TCP packets by setting the *FIN*, *PSH*, and *URG* flags, lighting the packet up like a Christmas tree [14]

After launching the attacks and giving some time for the background traffic to be generated for a reasonable period of time (around two hours), we downloaded a snapshot of the dataset of our experiment. The dataset snapshot is also available for download [19]. In order to validate that the dataset Algorizmi generated was not malformed, we ran the dataset against Snort. We then analyzed the alert files Snort generated that contain information about any malicious activity that Snort found while mining our sample dataset for possible intrusions. Snort was able to detect all four attacks successfully and generated alerts in the alert file for all of the generated attacks. Figure 5.2 shows the alerts for the **Xmas Scan**, while Figure 5.3 shows the alerts for the **Teardrop** attack, **Trin00** attack and the **UDP Port Scan** attack. We also noticed that the Nessus scan that we performed for all the VMIs in our network generated some traffic that was marked by Snort as malicious since we performed the most aggressive scan that Nessus can do against a machine by enabling all its plugins to be used during the scan. Figure 5.4 shows some of the alerts our Nessus scan generated. All Snort alert files are available for download [20].



```
alert
[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:52:28.108054 128.111.45.37:61065 -> 128.111.45.40:53
TCP TTL:41 TOS:0x0 ID:52416 IpLen:20 DgmLen:40
**U*P**F Seq: 0x96119F44 Ack: 0x0 Win: 0x000 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:52:29.209080 128.111.45.37:61065 -> 128.111.45.40:992
TCP TTL:43 TOS:0x0 ID:9117 IpLen:20 DgmLen:40
**U*P**F Seq: 0x96119F44 Ack: 0x0 Win: 0x1000 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:53:25.764272 128.111.45.37:58052 -> 128.111.45.40:18182
TCP TTL:46 TOS:0x0 ID:22280 IpLen:20 DgmLen:40
**U*P**F Seq: 0xC2CB81F9 Ack: 0x0 Win: 0xC00 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:53:26.867253 128.111.45.37:58052 -> 128.111.45.40:221
TCP TTL:43 TOS:0x0 ID:58193 IpLen:20 DgmLen:40
**U*P**F Seq: 0xC2CB81F9 Ack: 0x0 Win: 0x1000 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:55:49.609605 128.111.45.37:43679 -> 128.111.45.40:1139
TCP TTL:58 TOS:0x0 ID:46977 IpLen:20 DgmLen:40
**U*P**F Seq: 0x7914626 Ack: 0x0 Win: 0xC00 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:55:50.712620 128.111.45.37:43679 -> 128.111.45.40:1402
TCP TTL:49 TOS:0x0 ID:20120 IpLen:20 DgmLen:40
**U*P**F Seq: 0x7914626 Ack: 0x0 Win: 0x800 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-19:55:53.713722 128.111.45.37:59396 -> 128.111.45.40:443
TCP TTL:53 TOS:0x0 ID:52695 IpLen:20 DgmLen:40
**U*P**F Seq: 0x4A2A4B57 Ack: 0x0 Win: 0x000 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
```

Figure 5.2: Snort alerts for the Xmas scan

```
alert
**0P*** Seq: 0x176775F8 Ack: 0x0 Win: 0x400 TcpLen: 20 UrgPtr: 0x0
[Xref => http://www.whitehats.com/info/IDS30]

[**] [113:2:1] (spp_frag2) Teardrop attack [**]
12/23-20:25:44.140505 128.111.45.37 -> 128.111.45.40
UDP TTL:10 TOS:0x0 ID:242 IpLen:20 DgmLen:24
Frag Offset: 0x0003 Frag Size: 0x0004

[**] [1:246:2] DDOS nstream agent pong to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
12/23-20:25:44.933279 128.111.45.37:1779 -> 128.111.45.40:10498
UDP TTL:10 TOS:0x0 ID:13366 IpLen:20 DgmLen:32
Len: 4

[**] [1:245:3] DDOS nstream handler ping to agent [**]
[Classification: Attempted Denial of Service] [Priority: 2]
12/23-20:25:44.961436 128.111.45.37:1756 -> 128.111.45.40:10498
UDP TTL:10 TOS:0x0 ID:13366 IpLen:20 DgmLen:32
Len: 4
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0138]

[**] [1:244:3] DDOS nstream handler to agent [**]
[Classification: Attempted Denial of Service] [Priority: 2]
12/23-20:25:44.989442 128.111.45.37:1729 -> 128.111.45.40:10498
UDP TTL:10 TOS:0x0 ID:13366 IpLen:20 DgmLen:35
Len: 7
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0138]

[**] [1:243:2] DDOS nstream agent to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
12/23-20:25:45.017398 128.111.45.37:1682 -> 128.111.45.40:6838
UDP TTL:10 TOS:0x0 ID:13366 IpLen:20 DgmLen:37
Len: 9

[**] [1:223:3] DDOS Trin00 Daemon to Master PONG message detected [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-20:25:45.065548 128.111.45.37:1679 -> 128.111.45.40:31335
UDP TTL:10 TOS:0x0 ID:13366 IpLen:20 DgmLen:32
Len: 4
[Xref => http://www.whitehats.com/info/IDS187]

[**] [122:17:0] (partscan) UDP Portscan [**]
12/23-20:25:45.121487 128.111.45.37 -> 128.111.45.40
PROT0255 TTL:0 TOS:0x0 ID:13366 IpLen:20 DgmLen:165

[**] [1:239:2] DDOS shaft handler to agent [**]
[Classification: Attempted Denial of Service] [Priority: 2]
12/23-20:25:46.189375 128.111.45.37:1774 -> 128.111.45.40:18753
UDP TTL:10 TOS:0x0 ID:64289 IpLen:20 DgmLen:39
Len: 11
[Xref => http://www.whitehats.com/info/IDS255]
```

Figure 5.3: Snort alerts for Teardrop, Trin00 and UDP port scan attacks

```
alert
[Classification: Attempted Information Leak] [Priority: 2]
12/23-20:26:15.862470 128.111.45.37:1582 -> 128.111.45.40:162
UDP TTL:10 TOS:0x0 ID:15825 IpLen:20 DgmLen:46
Len: 18
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://
www.securityfocus.com/bid/4132][Xref => http://www.securityfocus.com/bid/
4089][Xref => http://www.securityfocus.com/bid/4088]

[**] [1:1411:10] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-17:02:53.942527 129.97.193.96:49406 -> 128.111.45.40:161
UDP TTL:49 TOS:0x0 ID:2909 IpLen:20 DgmLen:61
Len: 33
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://
cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0517][Xref => http://
www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/
4088][Xref => http://www.securityfocus.com/bid/2112]

[**] [1:1417:9] SNMP request udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-17:02:53.942527 129.97.193.96:49406 -> 128.111.45.40:161
UDP TTL:49 TOS:0x0 ID:2909 IpLen:20 DgmLen:61
Len: 33
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://
www.securityfocus.com/bid/4132][Xref => http://www.securityfocus.com/bid/
4089][Xref => http://www.securityfocus.com/bid/4088]

[**] [1:1411:10] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-17:02:55.943063 129.97.193.96:49407 -> 128.111.45.40:161
UDP TTL:49 TOS:0x0 ID:24680 IpLen:20 DgmLen:61
Len: 33
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://
cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0517][Xref => http://
www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/
4088][Xref => http://www.securityfocus.com/bid/2112]

[**] [1:1417:9] SNMP request udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-17:02:55.943063 129.97.193.96:49407 -> 128.111.45.40:161
UDP TTL:49 TOS:0x0 ID:24680 IpLen:20 DgmLen:61
Len: 33
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://
www.securityfocus.com/bid/4132][Xref => http://www.securityfocus.com/bid/
4089][Xref => http://www.securityfocus.com/bid/4088]
```

Figure 5.4: Snort alerts generated by the traffic of our Nessus scan

Requirement	Is Satisfied ?	Requirement	Is Satisfied ?
<b>Availability</b>		<b>Measurement Tools</b>	
Publicly available	✓	Trace collection tools	✓
Open-source	✓	Non-disruptive running of tools	✓
<b>Versatility</b>		<b>Datasets</b>	
Configurable environment	✓	Shared datasets	✓
High-bandwidth network emulation	-	Documented datasets	✓
Different network models	✓	Different Output formats	✓
<b>Background Traffic Fidelity</b>		<b>Containment and Security</b>	
Tuning	✓	Isolation from the Internet	✓
Pre-set models	✓	Data protection	✓
Add new models	✓	<b>Usability</b>	
Extract model from available traces	-	Smooth deployment	✓
Models database	✓	Testbed reconfiguration	✓
<b>Attack Traffic Fidelity</b>		Full control on testbed	✓
Attack selection	✓	Scalable testbed	✓
Attack-free period	✓	User friendly interface	✓
Updateable database	✓	<b>Reproducibility and Maintenance</b>	
Different attacking machines	✓	Save experiment configuration	✓
<b>Economy and Efficiency</b>		Resource replacement	✓
Low cost	✓		
Resource utilization	✓		

Table 5.1: Compliance of Algorizmi to the typical research requirements

## 5.4 Design Requirements Satisfaction

Algorizmi is an open-source project and is publicly available for use by the research community [2]. Thus, it satisfies the availability requirements. For the versatility requirement, Algorizmi offers a configurable environment for its users. In addition, users can have different network models in their experiments. This is true in the case that we are not using EPC or Amazon EC2 as the cloud infrastructure for Algorizmi as this option needs to modify the default settings of Algorizmi’s Eucalyptus-based cloud to allow the user to modify the IP settings of the VMIs she instantiates. However, current physical resources do not allow Algorizmi to emulate high-bandwidth ( $\geq 1$ Gbs) networks. That is the reason why Algorizmi does not fully satisfy the design requirements of offering a configurable environment and providing Algorizmi users with the ability to build various network models.

LARIAT [107] is equipped with tools that can extract background traffic models from a given network traffic trace. On the other hand, Algorizmi lacks such tools, but has the



ability to generate attack traffic from multiple machines to avoid favoring a specific IDS algorithm that is being evaluated using the generated dataset.

None of the studied research works estimate how the cost of implementing their systems can be, or how they are trying to achieve maximum utilization of their resources except for DETER [35] and LassNetExp [93]. Algorizmi depends in its physical infrastructure on commodity hardware to host its cloud infrastructure and its database. In addition, all the libraries used by Algorizmi are open-source software packages that are free to use. The cloud infrastructure of Algorizmi can be any Amazon EC2 interface-compatible cloud, but we opted for Eucalyptus since it is the open-source option we found. Having a cloud infrastructure that consists of multiple clusters will achieve the maximum utilization of the given resources as opposed to running Algorizmi on one physical node.

Algorizmi supports multiple output formats (more than 20 different formats) for the generated dataset, all of which are readable by the widely used network protocol analyzer Wireshark [27]. This feature puts Algorizmi ahead of other research projects since none of them supports different output formats for the generated dataset. Table 5.1 summarizes the compliance of Algorizmi to each of the design requirements presented in Chapter 3, compared to other research projects.

## 5.5 Performance Assessment

### 5.5.1 Environment Settings

We deployed the Algorizmi Desktop Application on a Macbook laptop that has 4 GB of RAM, with a 2.4 GHz Intel Core 2 Duo processor, and running Mac OS X 10.5.8. Algorizmi Desktop Application is a Java-based desktop application built as a GUI for Algorizmi to allow its users to interact with the system.

### 5.5.2 Performance Metrics

We evaluate the performance of Algorizmi based on five metrics:

1. **CPU Usage:** how much of the CPU power (as a percentage of the total CPU power) of the client Algorizmi Desktop Application is deployed on, is consumed on average.
2. **CPU Time:** the average CPU time (in seconds) Algorizmi Desktop Application takes to perform a task.

3. **Memory Usage:** the average amount of KBs of RAM used by the Algorizmi Desktop Application to perform a certain task.
4. **Bytes Sent/Received:** the average amount of network traffic sent/received by the client running the Algorizmi Desktop Application measured in KBs.
5. **Execution Time:** the average time taken by the Algorizmi Desktop Application to serve a request initiated by the user.

### 5.5.3 Results

For each of the most common tasks a user can perform using Algorizmi, we execute this task multiple times (ranging from 8 to 15 times for each task, depending on how time-consuming the task is) and we record the average values of the parameters we chose to evaluate Algorizmi upon. Table 5.2 shows that the Algorizmi Desktop uses at most 0.31% of the CPU to execute a given task. Results drawn from Table 5.2 and represented in Figure 5.5 also show that the relation between the number of VMIs created at once and the time taken to execute such task is almost linear. On the other hand, some other results seems to be bizzare. For example, adding one VMI to an experiments consume more bytes (sent and received) than adding two VMIs. This is weird since both requests sent to Algorizmi are exactly the same with one difference in the number of requested VMIs. Firstly, we attributed this behavior to a traffic congestion that took place while sending the first request that led to re-sending the request several times until it reached Algorizmi. However, this behavior was repeatable whenever we performed this task even on different times of the day.

## 5.6 Summary

In this chapter, we compared Algorizmi to previous research projects that tried to solve the problem of IDS evaluation. Algorizmi was found to satisfy more research needs and requirements than the other projects. We also assessed the performance of Algorizmi while executing some of the most common tasks that a typical user can do using the Algorizmi Desktop Application.

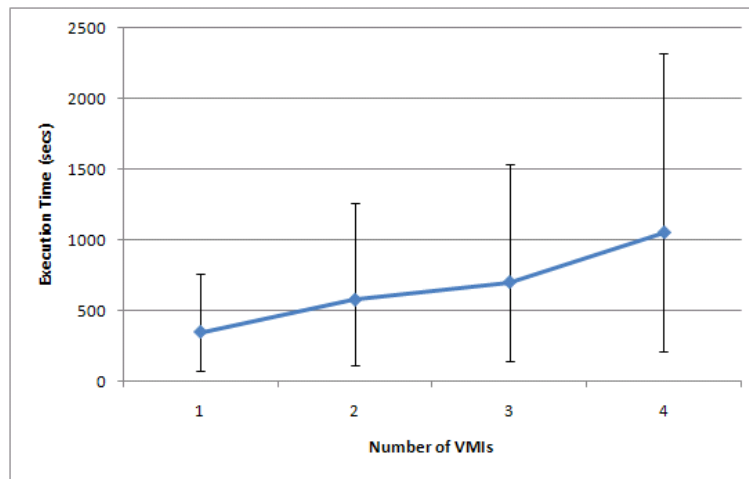


Figure 5.5: Response time to user request for instantiating a set of VMs

	CPU Usage (%)	CPU Time (secs)	Memory Usage (KBs)	Bytes Sent (KBs)	Bytes Received (KBs)	Execution Time (secs)
Creating a new experiment	0.10	0.71	1030.80	3.11	2.10	7.24
Creating a new SSH key pair	0.13	0.86	2219.27	1.89	3.58	6.75
Deleting an experiment	0.23	0.19	744.00	2.23	3.78	0.95
Reporting available resources	0.31	2.87	7128.22	11.78	76.28	18.59
Adding a VMI to an experiment	0.06	20.44	1701.33	638.72	18194.85	346.18
Adding 2 VMIs to an experiment	0.06	31.62	1118.67	490.95	7287.11	575.04
Adding 3 VMIs to an experiment	0.05	38.15	6946	3979.24	42933.16	700.09
Adding 4 VMIs to an experiment	0.05	57.83	112541.71	1348.36	13531.48	1058.85

Table 5.2: Algorithm Performance Assessment

# Chapter 6

## Conclusion

### 6.1 Summary of Contributions

This thesis has two major contributions: surveying previous research work done in the field of IDS evaluation, and proposing an open-source configurable virtual testbed for dataset generation for offline evaluation of IDSes.

#### 6.1.1 Survey of IDS Evaluation Systems

Our survey of the previous research efforts done in the field of IDS offers an overview of the challenges and problems faced. We, then, provide a better understanding of those challenges to help us determine what problems are more crucial than others. Consequently, we found that one of the most important open research problems is how to satisfy the research community requirements regarding an IDS evaluation system. In specific, the need for a configurable tool to generate datasets that can be used for offline evaluation of IDSes. Finally, we surveyed various research projects that tried to solve this problem and based on which we compiled and categorized the design requirements that we believe should be present in a system used for generating datasets for IDS offline evaluation.

#### 6.1.2 Algorizmi

We designed Algorizmi based on the specified design requirements. An architecture of Algorizmi and its detailed system components are proposed in this thesis. In addition, we compared Algorizmi with previous research work regarding the compliance of each project to the design requirements. The result shows that Algorizmi satisfies almost all

the requirements (28 out of 30) which is far more than any other project did. Algorizmi allows users to configure their experiments to include different network models, generate various background traffic, launch a wide variety of attacks against their networks. In addition, a user can download the dataset of her experiment in various output formats. Not only is Algorizmi publicly available for use, its source code is also publicly shared [2] so that the whole community can contribute to. We showed how Algorizmi was able to generate a dataset that was fed to Snort which successfully detected the attacks represented in the dataset. Finally, we evaluated the performance of Algorizmi while responding to the most common tasks a user can do while using the system.

## 6.2 Limitations

### 6.2.1 Using EPC

Using the Eucalyptus Public Cloud (EPC) [7] imposed some limitations on Algorizmi's Eucalyptus-based cloud infrastructure due to EPC's Service Level Agreement (SLA):

1. VMIs are time-limited to 6 hours (they are terminated without any warning).
2. No more than 4 VMIs from one user are allowed at any time.
3. VMIs are assigned public IP addresses, but only incoming network connections and inter-instance network connections are allowed.
4. A user can have a maximum of 5GB of permanent storage on EPC.

Additional limitations are inherently imposed on Algorizmi due to using EPC. Since we are not the administrators of this public cloud, we could not configure the cloud so that we can expand the range of requirements Algorizmi could satisfy. In other words, a Eucalyptus-based cloud (including the EPC) can operate in four different networking modes depending on the desired level of functionality. Each VMI in any of those modes gets assigned by the cloud controller two IP addresses:

1. **a private IP address:** used within the private network of the cluster (i.e availability zone) this instance belongs to.
2. **a public IP address:** used by the owner of the VMI to interact with it from outside the cloud.

The EPC does not allow its users to manipulate those assigned IP addresses which prevents an Algorizmi user from creating a network topology of her choice for her experiment. This limitation can be overcome by building our own Eucalyptus-based cloud with our own physical resources. As admins, we can then configure the cloud to allow users to set the private addresses of their VMIs which basically gives them unlimited choices of network topologies they can use for their experiments.

### **6.2.2 Metasploit Integration**

Although Metasploit serves its purpose of being an up-to-date customizable source of attacks scripts, it does not have an API. Therefore, we were not able to fully integrate Metasploit within Algorizmi. This causes a user of Algorizmi to run the attacks against her experimental testbeds using Metasploit independently. Had we managed to integrate Metasploit with Algorizmi, the overall process could have been more smooth and user-friendly. However, we can extend the design of Algorizmi to include an API for Metasploit programmability, whether we develop this API or Metasploit releases one.

## **6.3 Future Work**

Algorizmi can be extended in various ways in order to provide its users with a richer experience and increase the set of features it provides to them. Firstly, we can build our own Eucalyptus-based cloud infrastructure so that we relieve Algorizmi from the limitations imposed on it due to using the EPC. Secondly, we can develop an API for Metasploit so that we can fully integrate it with the design of Algorizmi to offer the users a simpler way to generate network attacks against their experimental testbeds. Thirdly, we can extend the notion of templates in Algorizmi to include experiment templates (in addition to the existing background script templates). We can offer, and users can create and share if they want to, some already-made designs for experiments that can be used by Algorizmi users. This way, the process of building an experiment (i.e virtual testbed) will be much easier. Fourthly, Algorizmi can offer the users tools that facilitate the process of configuring, and building virtual machines, each with its own service and application stacks. Finally, Algorizmi can turn into an environment used for online IDS evaluation rather than a system that generates datasets of virtual networks that are eventually used to evaluate IDS algorithms offline. Such an environment should allow a user to deploy her own IDS on her experiment and evaluate it under synthetic conditions that she configures.

## 6.4 Concluding Note

We presented in this thesis Algorizmi, an open-source configurable virtual testbed to generate datasets for offline evaluation of IDSes. Although there are still some issues that need to be addressed, Algorizmi satisfies more research needs than previous similar research projects. We believe that there is a high potential that motivates the work on Algorizmi and its development. As a result, we anticipate that researchers will have the option of evaluating their IDS algorithms without any worries about how they will be able to configure their testbed, reproduce their results if desired or share their results with other researchers.



# APPENDICES

# Appendix A

## Algorizmi Implementation Experience

This appendix gives an overview of some lessons learned during the implementation process. These can serve as a guide to other researchers if they design a similar system or build on Algorizmi. At first, our goal was to build our own Eucalyptus-based cloud so that we would be able to adjust the networking mode of the cloud to allow users to assign private IP addresses to the VMIs allowing them to have different network topologies within their experiments. We made multiple trials to achieve that goal. Firstly, we tried to install the Eucalyptus-based cloud on the laptop machine we had since it has more resources than the desktop machine we own. We tried to find a binary package for Eucalyptus for Mac OS X (the OS running on the laptop) to install, but we could not. Therefore, we had to resort to installation from source.

Afterwards, we searched for many dependencies for Eucalyptus and install from source as well because all of the dependencies did not have a binary file for Mac OS X. We faced many problems due to the 32-bit/64-bit compatibility issues. After installing most of the dependencies of Eucalyptus, we got stuck at one specific package that had no support for Mac OS X. After mailing the development team, they confirmed that they do not support Mac OS X so we tried to seek help from the development team of Eucalyptus and they suggested that we install Eucalyptus from its source on a Ubuntu virtual machine on our laptop machine.

We used Sun Virtualbox (an open-source virtualization tool) to create the Ubuntu virtual machine on our laptop machine. Installing Eucalyptus on any machine (even a virtual one) requires that this machine support either Xen or KVM (Kernel-based Virtual Machine) so that Eucalyptus will be able to instantiate virtual machine instances within the cloud infrastructure. We opted for using KVM since it was the recommended setting for installing Eucalyptus Ubuntu. It took us quite a while until we figured out that although

our laptop supports working with KVM, somehow Virtualbox was not able to emulate such capability and pass it to the VM we installed as it is supposed to.

Our second option was to abandon KVM and install a xen-enabled kernel on the Ubuntu VM so that we can successfully install Eucalyptus. This step alone took us some time before we found a xen-enabled kernel that is compatible with Ubuntu 9.04 (the latest version of Ubuntu then). The problem is that Ubuntu stopped shipping xen-enabled kernels (in favor of the new KVM technology) since Ubuntu 8.10. Finally, we managed to install the xen-enabled kernel on my Ubuntu VM, however the VM was not able to boot from it. This behavior was attributed to the impracticality of nested virtualization when dealing with hypervisors. However, this is not a problem when one is using front-end virtualization (e.g. installing Virtualbox on a VM that is hosted by a Virtualbox installation). Since all our attempts to install Eucalyptus on our laptop machine failed. Finally, we opted for installing it on our desktop machine.

Our experience with installing Eucalyptus on our desktop machine was as difficult as installing it on our laptop. We spent quite some time debugging an error that was raised by Eucalyptus to reach its root cause. The root cause turned out to be that the CPU of our desktop machine does not support KVM so we had to install Xen and go through the same process that we did with our laptop machine.

A major difficulty that we faced was that for Eucalyptus to give access to the VMIs from outside the Eucalyptus cloud (i.e. any connection to the instances from outside the cloud), it should assign public IPs to those VMs. This required that our Eucalyptus controller node to be able to contact a DHCP server and lease IP addresses to those VMs. This is not possible to achieve in the small setup I had in the office (only one machine working) since we do not have access to any DHCP server that will allow this kind of behavior. Moreover, we cannot use the DHCP server of the CS servers (for example) since they have to know the MAC address that will be associated with the leased IP which is impractical since VMIs are initialized with randomly generated MAC addresses. A workaround was to use NATing but we were not quite sure about the legality of such issue so we opted for using the Eucalyptus Public Cloud (EPC) instead of our own Eucalyptus cloud although this incurred some limitations as discussed earlier.

At this point, we used to run our application by running Eucalyptus command lines from within the application. Therefore, we had to start searching for a Java API to be used with Eucalyptus to have more flexibility dealing with the cloud platform. Searching many resources on the Internet, we found a Java API for Amazon EC2 developed by Amazon and since Eucalyptus is interface-compatible with Amazon EC2, we were able to use this library to interact with EPC.

We were left with another challenge: integrating Metasploit with Algorizmi. Metasploit was written in Ruby and it does not have any kind of APIs (even a Ruby API) so we had

to write some ruby scripts to be able to do some of the functionalities of Metasploit from within our Java application (namely enumerating attacking modules, fetching info about each module). However, this method was not flexible enough to allow me to develop scripts (runnable from Algorizmi) to run the attacks from Metasploit because this means that we should develop scripts for all the modules in Metasploit which is not a practical solution. Therefore, Metasploit is not integrated within Algorizmi and the user has to run Metasploit by itself in order to launch attacks from it against her experimental testbed.

Finally, after successfully running multiple experiments that had nodes generating both background traffic and attack traffic, we found out that the EPC admins blocked our account. We checked with them and explained that we were doing research work (the work contributing to this Master's thesis) and they allowed us to install and use the required tools to generate the background traffic. However, they blocked us from generating any attack traffic or even downloading Metasploit on the VMIs. Their reply was:

“We do not open up port for individual projects. The EPC is there to drive test a cloud and check what it can do and opening up extra port would require us to start monitoring what is running inside the EPC (something we don't want to do).”

# References

- [1] 10th CSI/FBI survey shows dramatic increase in unauthorized access. *IT Professional*, 7(4):1–23, July-Aug. 2005. 1
- [2] Algorizmi. <http://sourceforge.net/projects/algorizmi/>, Dec. 2009. 60, 66
- [3] Amazon EC2 Query Interface. <http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/query-apis.html>, Jan. 2009. 30
- [4] Amazon EC2 Soap Interface. <http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/soap-apis.html>, Jan. 2009. 30
- [5] Amazon Elastic Compute Cloud (Amazon EC2). <http://aws.amazon.com/ec2/>, Jan. 2009. 28, 29, 30
- [6] Amazon Simple Storage Service (S3), Jan. 2009. 28
- [7] Eucalyptus Public Cloud. <http://open.eucalyptus.com/wiki/EucalyptusPublicCloud>, Jul. 2009. 29, 66
- [8] The Expect Home Page. <http://expect.nist.gov/>, Nov. 2009. 36
- [9] An Introduction To Unix Shell Programming, Nov. 2009. 36
- [10] Java Library for Amazon EC2, Jun. 2009. 31
- [11] Metasploit Framework, Jan. 2009. 26, 38
- [12] Nessus. <http://www.nessus.org/nessus/>, Dec. 2009. 56
- [13] Nessus vulnerability report for a sample network built in algorizmi. <http://www.cs.uwaterloo.ca/~karim/thesis/VulnerabilityScan-Nessus.html.zip>, Dec. 2009. 56
- [14] Nmap Xmas Scan. <http://nmap.org/book/man-port-scanning-techniques.html/>, Dec. 2009. 56

- [15] The Perl Directory, Nov. 2009. 36
- [16] PHP: Hypertext Preprocessor. <http://php.net/index.php>, Nov. 2009. 36
- [17] phpbb - Creating Communities Worldwide. <http://www.phpbb.com/?sid=fcf07b0c40f532a4bc761ce5cc2dc9f7>, Dec. 2009.
- [18] Python Programming Language. <http://www.python.org/>, Nov. 2009. 36
- [19] Sample dataset generated by Algorizmi. <http://www.cs.uwaterloo.ca/~karim/thesis/dataset.zip>, Dec. 2009. 56
- [20] Snort alert files for a sample dataset generated by Algorizmi. <http://www.cs.uwaterloo.ca/~karim/thesis/SnortAlerts.zip>, Dec. 2009. 56
- [21] tcpdump/libpcap, March 2009. 26
- [22] Teardrop Attacks. [http://en.wikipedia.org/wiki/Teardrop\\_Attacks](http://en.wikipedia.org/wiki/Teardrop_Attacks), Dec. 2009. 56
- [23] Trin00. <http://en.wikipedia.org/wiki/Trinoo>, Dec. 2009. 56
- [24] tshark. <http://www.wireshark.org/docs/man-pages/tshark.html>, Jul. 2009. 43, 55
- [25] UDP Port Scan. [http://en.wikipedia.org/wiki/Port\\_scanner#UDP\\_scanning](http://en.wikipedia.org/wiki/Port_scanner#UDP_scanning), Dec. 2009. 56
- [26] Vulnerability Management - Rapid7. <http://www.rapid7.com/>, Nov. 2009. 38
- [27] Wireshark. <http://www.wireshark.org/>, Nov. 2009. 43, 61
- [28] Mustaque Ahamad, Dave Amster, Michael Barrett, Tom Cross, George Heron, Don Jackson, Jeff King, Wenke Lee, Ryan Naraine, Gunter Ollmann, Jon Ramsey, Howard A. Schmidt, and Patrick Traynor. Emerging cyber threats report for 2009. White Paper, Georgia Tech Information Security Center, 2009. 1
- [29] Mark Allman and Vern Paxson. Issues and etiquette concerning use of shared measurement data. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 135–140, New York, NY, USA, 2007. ACM. 9
- [30] Demetris Antoniadis, Panagiotis Trimintzios, Michalis Polychronakis, Sven Ubik, Antonis Papadogiannakis, and Vladimir Smotlacha. Lobster: a european platform for passive network traffic monitoring. In *TridentCom '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development*

- of networks & communities*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [31] N. Athanasiades, R. Abler, J. Levine, H. Owen, and G. Riley. Intrusion detection testing and benchmarking methodologies. *Information Assurance, 2003. IWIAS 2003. Proceedings. First IEEE International Workshop on*, pages 63–72, March 2003. [8](#), [9](#)
  - [32] Stefan Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 1–7, New York, NY, USA, 1999. ACM. [2](#)
  - [33] Naimul Basher, Aniket Mahanti, Anirban Mahanti, Carey Williamson, and Martin Arlitt. A comparative analysis of web and peer-to-peer traffic. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 287–296, New York, NY, USA, 2008. ACM.
  - [34] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with deter: a testbed for security research. *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, pages 10 pp.–388, March 2006. [11](#)
  - [35] Terry Benzel, Robert Braden, E. Fraser, Anthony Joseph, Dongho Kim, J. Mehringer, Clifford Neuman, Ron Ostrenga, and Stephen Schwab and Dan Sterne. Design of the deter security testbed. Technical report, USC Information Sciences Institute, University of California at Berkeley and McAfee Research, May 2004. [5](#), [11](#), [13](#), [17](#), [19](#), [20](#), [21](#), [61](#)
  - [36] Josep L. Berral, Nicolas Poggi, Javier Alonso, Ricard Gavaldà, Jordi Torres, and Manish Parashar. Adaptive distributed mechanism against flooding network attacks based on machine learning. In *AISec '08: Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 43–50, New York, NY, USA, 2008. ACM.
  - [37] Giuseppe Bianchi, Simone Teofili, and Matteo Pomposini. New directions in privacy-preserving anomaly detection for network traffic. In *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, pages 11–18, New York, NY, USA, 2008. ACM. [5](#), [9](#), [13](#)
  - [38] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164, New York, NY, USA, 2006. ACM.

- [39] Martin Burkhart, Daniela Brauckhoff, Martin May, and Elisa Boschi. The risk-utility tradeoff for ip address truncation. In *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, pages 23–30, New York, NY, USA, 2008. ACM.
- [40] C. Caini, R. Firrincieli, D. Lacamera, S. Tamagnini, and D. Tiraferri. The tatpa testbed; a testbed for advanced transport protocols and architecture performance evaluation on wireless channels. *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*, pages 1–7, May 2007.
- [41] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 99–106, Washington, DC, USA, 2005. IEEE Computer Society. 7
- [42] Henri Casanova. Simgrid: A toolkit for the simulation of application scheduling. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:430, 2001. 7
- [43] Charlie Catlett. The philosophy of teragrid: Building an open, extensible, distributed terascale facility. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:8, 2002. 7
- [44] Dong Chen, Hanping Hu, Zuxi Wang, and Jianghang Chen. A novel method for network anomaly detection using superstatistics. *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, pages 595–598, March 2008. 10
- [45] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003. 7
- [46] P. Cortez, M. Rio, M. Rocha, and P. Sousa. Internet traffic forecasting using neural networks. *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 2635–2642, July 2006.
- [47] R. Crawford, M. Bishop, B. Bhumiratana, L. Clark, and K. Levitt. Sanitization models and their limitations. In *NSPW '06: Proceedings of the 2006 workshop on New security paradigms*, pages 41–56, New York, NY, USA, 2007. ACM.
- [48] Hervé Debar, Marc Dacier, Andreas Wespi, and Stefan Lampart. A workbench for intrusion detection systems. Technical Report RZ 6519, IBM Zurich Research Laboratory, Switzerland, March 1998. 5, 9, 13



- [49] Guang-Yu Du, Tian-Shu Huang, Bing-Jie Zhao, and Li-Xin Song. Dynamic self-defined immunity model base on data mining for network intrusion detection. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 6:3866–3870, Aug. 2005.
- [50] E. Eskin, Wenke Lee, and S.J. Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings*, 1:165–175, 2001. 10
- [51] J.M. Estevez-Tapiador, P. Garcia-Teodoro, and J.E. Diaz-Verdejo. Stochastic protocol modeling for anomaly based network intrusion detection. *Information Assurance, 2003. IWIAS 2003. Proceedings. First IEEE International Workshop on*, pages 3–12, March 2003.
- [52] Kevin Fall. Network emulation in the vint/ns simulator. *Computers and Communications, IEEE Symposium on*, 0:244, 1999. 7
- [53] S. Floyd and V. Paxson. Difficulties in simulating the internet. *Networking, IEEE/ACM Transactions on*, 9(4):392–403, Aug 2001. 7
- [54] Marc Fossi, Eric Johnson, Trevor Mack, Dean Turner, Joseph Blackbird, Mo King Low, Teo Adams, David McKinney, Stephen Entwisle, Marika Pauls Laucht, Candid Wueest, Paul Wood, Dan Bleaken, Greg Ahmad, Darren Kemp, and Ashif Samnani. Symantec global internet security threat report. White Paper, Symantec Enterprise Security, April 2009. 1
- [55] Carrie E. Gates. A case study in testing a network security algorithm. In *TridentCom '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, pages 1–6, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [56] Shantanu Gattani and Thomas E. Daniels. Reference models for network data anonymization. In *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, pages 41–48, New York, NY, USA, 2008. ACM.
- [57] Joshua W. Haines, Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. 1999 darpa intrusion detection system evaluation: Design and procedures. Technical report, MIT Lincoln Laboratory, Lexington, Massachusetts, USA, February 2001. 5, 8, 9, 10, 13
- [58] J.W. Haines, L.M. Rossey, R.P. Lippmann, and R.K. Cunningham. Extending the darpa off-line intrusion detection evaluations. *DARPA Information Survivability*

- Conference & Exposition II, 2001. DISCEX '01. Proceedings*, 1:35–45, 2001. 16, 17, 21
- [59] Sang-Jun Han and Sung-Bae Cho. Evolutionary neural networks for anomaly detection based on the behavior of a program. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 36(3):559–570, June 2005. 10
- [60] L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 296–304, May 1990. 9
- [61] A. Holland and S.J. Lincke. Configuring a lab to support community-based security audit projects. *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*, pages 1–7, May 2007.
- [62] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile, 1999. 28
- [63] Alefiya Hussain, Genevieve Bartlett, Yuri Pryadkin, John Heidemann, Christos Papadopoulos, and Joseph Bannister. Experiences with a continuous network tracing infrastructure. In *MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 185–190, New York, NY, USA, 2005. ACM.
- [64] Gianluca Iannaccone, Christophe Diot, Derek McAuley, Andrew Moore, Ian Pratt, and Luigi Rizzo. The como white paper. Technical report, Intel Research, Cambridge, UK, September 2004.
- [65] Gianluca Iannaccone, Christophe Diot, Ian Graham, and Nick McKeown. Monitoring very high speed links. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 267–271, New York, NY, USA, 2001. ACM.
- [66] Marios Iliofotou, Prashanth Pappu, Michalis Faloutsos, Michael Mitzenmacher, Sumeet Singh, and George Varghese. Network monitoring using traffic dispersion graphs (tdgs). In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 315–320, New York, NY, USA, 2007. ACM.
- [67] Shu-Yuan Jin, D.S. Yeung, and Xi-Zhao Wang. A second-order statistical detection approach with application to internet anomaly detection. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 5:3260–3264, Aug. 2005. 10

- [68] Dae-Ki Kang, D. Fuller, and V. Honavar. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 118–125, June 2005. [10](#)
- [69] A. Karygiannis, K. Robotis, and E. Antonakakis. Creating offline manet ids network traces. *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6, June 2007.
- [70] Michelle Keeney, Dawn Cappelli, Eileen Kowalski, Andrew Moore, Timothy Shimeall, and Stephanie Rogers. Insider threat study: computer system sabotage in critical infrastructure sectors. Technical report, U.S.S. Service and C.M.U. Software Engineering Institute, May 2005. [3](#)
- [71] Michael S. Keller. Take command: cron: Job scheduler. *Linux J.*, page 15. [42](#)
- [72] Douglas J. Kelly, Richard A. Raines, Michael R. Grimaila, Rusty O. Baldwin, and Barry E. Mullins. A survey of state-of-the-art in anonymity metrics. In *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, pages 31–40, New York, NY, USA, 2008. ACM. [5](#), [8](#), [9](#), [13](#)
- [73] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master’s thesis, Massachusetts Institute of Technology, June 1999. [10](#)
- [74] Seong Soo Kim and A.L.N. Reddy. Image-based anomaly detection technique: Algorithm, implementation and effectiveness. *Selected Areas in Communications, IEEE Journal on*, 24(10):1942–1954, Oct. 2006.
- [75] Marius Kloft, Ulf Brefeld, Patrick Düessel, Christian Gehl, and Pavel Laskov. Automatic feature selection for anomaly detection. In *AISec '08: Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 71–76, New York, NY, USA, 2008. ACM.
- [76] Stefan Kornexl, Vern Paxson, Holger Dreger, Anja Feldmann, and Robin Sommer. Building a time machine for efficient recording and retrieval of high-volume network traffic. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 23–23, Berkeley, CA, USA, 2005. USENIX Association.
- [77] Kiran Lakkaraju and Adam Slagell. Evaluating the utility of anonymized network traces for intrusion detection. In *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication networks*, pages 1–8, New York, NY, USA, 2008. ACM.

- [78] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. *SIGMETRICS Perform. Eval. Rev.*, 34(1):145–156, 2006.
- [79] George S. Lee, Lachlan L. H. Andrew, Ao Tang, and Steven H. Low. Wan-in-lab: Motivation, deployment and experiments. In *In Proc. PFLDnet, pp 85-90, Marina Del*, pages 85–90, 2007. 7
- [80] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, 2000.
- [81] Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. *Machine Learning and Cybernetics, 2003 International Conference on*, 5:3077–3081, Nov. 2003. 10
- [82] Shu Yun Lim and Andy Jones. Network anomaly detection system: The state of art of network behaviour analysis. In *ICHIT '08: Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology*, pages 459–465, Washington, DC, USA, 2008. IEEE Computer Society.
- [83] Yong Liu, Don Towsley, Tao Ye, and Jean C. Bolot. An information-theoretic approach to network monitoring and measurement. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 14–14, Berkeley, CA, USA, 2005. USENIX Association.
- [84] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. Is sampled data sufficient for anomaly detection? In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 165–176, New York, NY, USA, 2006. ACM.
- [85] Frédéric Massicotte, François Gagnon, Yvan Labiche, Lionel Briand, and Mathieu Couture. Automatic evaluation of intrusion detection systems. *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, pages 361–370, Dec. 2006. 5, 11, 13
- [86] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, 2000. 5, 10, 11
- [87] Peter Mell, Vincent Hu, Richard Lippmann, Josh Haines, and Marc Zissman. An overview of issues in testing intrusion detection systems. Technical Report 7007, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, June 2003. 5, 9, 12, 13, 17

- [88] Jelena Mirkovic. Privacy-safe network trace sharing via secure queries. In *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, pages 3–10, New York, NY, USA, 2008. ACM.
- [89] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.
- [90] Jeffrey C. Mogul and Martin Arlitt. Sc2d: an alternative to trace anonymization. In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 323–328, New York, NY, USA, 2006. ACM. [5](#), [9](#), [13](#)
- [91] D. Nurmi, R. Wolski, C. Grzegorzcyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124–131, May 2009. [28](#)
- [92] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 151–156, New York, NY, USA, 2008. ACM.
- [93] Philippe Owezarski, Pascal Berthou, Yann Labit, and David Gauchard. Laasnetexp: a generic polymorphic platform for network emulation and experiments. In *Trident-Com '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). [xi](#), [5](#), [6](#), [8](#), [12](#), [13](#), [17](#), [20](#), [21](#), [61](#)
- [94] Ruoming Pang and Vern Paxson. A high-level programming environment for packet trace anonymization and transformation. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 339–351, New York, NY, USA, 2003. ACM.
- [95] Kihong Park and Walter Willinger. *Self-Similar Network Traffic and Performance Evaluation*. John Wiley & Sons, Inc., New York, NY, USA, 2000.
- [96] Animesh Pacha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470, 2007. [2](#), [5](#)
- [97] Vern Paxson. Bro: a system for detecting network intruders in real-time. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium*, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association. [1](#)

- [98] Vern Paxson. Strategies for sound internet measurement. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 263–271, New York, NY, USA, 2004. ACM.
- [99] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1):3, 2007.
- [100] Markus Peuhkuri. A method to compress and anonymize packet traces. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 257–261, New York, NY, USA, 2001. ACM.
- [101] Phillip Porras and Vitaly Shmatikov. Large-scale collection and sanitization of network security data: risks and challenges. In *NSPW '06: Proceedings of the 2006 workshop on New security paradigms*, pages 57–64, New York, NY, USA, 2007. ACM.
- [102] Himabindu Pucha, Y. Charlie Hu, and Z. Morley Mao. On the impact of research network based testbeds on wide-area experiments. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 133–146, New York, NY, USA, 2006. ACM.
- [103] Nicholas Puketza, Mandy Chung, Ronald A. Olsson, and Biswanath Mukherjee. A software platform for testing intrusion detection systems. *IEEE Softw.*, 14(5):43–51, 1997. 9
- [104] Hai Qiu, Eklund Neil, Xiao Hu, Weizhong Yan, and Iyer Naresh. Anomaly detection using data clustering and neural networks. *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 3627–3633, June 2008.
- [105] Anirudh Ramachandran, Srinivasan Seetharaman, Nick Feamster, and Vijay Vazirani. Fast monitoring of traffic subpopulations. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 257–270, New York, NY, USA, 2008. ACM.
- [106] Martin Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association. 1, 54
- [107] L.M. Rossey, R.K. Cunningham, D.J. Fried, J.C. Rabek, R.P. Lippmann, J.W. Haines, and M.A. Zissman. Lariat: Lincoln adaptable real-time information assurance testbed. *Aerospace Conference Proceedings, 2002. IEEE*, 6:6–2671–2676, 6–2678–6–2682, 2002. 5, 10, 13, 60

- [108] Javier Rubio-Loyola, Dolors Sala, and Ali Ismail Ali. Maximizing packet loss monitoring accuracy for reliable trace collections. *Local and Metropolitan Area Networks, 2008. LANMAN 2008. 16th IEEE Workshop on*, pages 61–66, Sept. 2008.
- [109] Andy Rupp, Holger Dreger, Anja Feldmann, and Robin Sommer. Packet trace manipulation framework for test labs. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 251–256, New York, NY, USA, 2004. ACM.
- [110] Antoine Scherrer, Nicolas Larrieu, Philippe Owezarski, Pierre Borgnat, and Patrice Abry. Non-gaussian and long memory statistical characterizations for internet traffic with anomalies. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):56–70, Jan.-March 2007.
- [111] Xiaoxin Shao, Qianli Zhang, Tao He, Shijin Kong, Changqin An, and Xing Li. Santt: Sharing anonymized network traffic traces among researchers. *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 527–533, April 2006. [5](#), [8](#), [9](#), [13](#)
- [112] Douglas C. Sicker, Paul Ohm, and Dirk Grunwald. Legal issues surrounding monitoring during network research. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 141–148, New York, NY, USA, 2007. ACM. [9](#)
- [113] Joel Sommers and Paul Barford. Self-configuring network traffic generation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 68–81, New York, NY, USA, 2004. ACM.
- [114] Joel Sommers, Vinod Yegneswaran, and Paul Barford. A framework for malicious workload generation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 82–87, New York, NY, USA, 2004. ACM.
- [115] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The microgrid: a scientific tool for modeling computational grids. *SC Conference*, 0:53, 2000. [7](#)
- [116] N. Srinivasan and V. Vaidehi. Reduction of false alarm rate in detecting network anomaly using mahalanobis distance and similarity measure. *Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on*, pages 366–371, Feb. 2007. [10](#)
- [117] Feixian Sun, Qiusheng Zheng, and Tao Li. Immunity-based dynamic anomaly detection method. *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 644–647, May 2008.

- [118] Atsuko Takefusa, Satoshi Matsuoka, Kento Aida, Hidemoto Nakada, and Umpei Nagashima. Overview of a performance evaluation system for global computing scheduling algorithms. *High-Performance Distributed Computing, International Symposium on*, 0:11, 1999. 7
- [119] Ming Tian, Song-Can Chen, Yi Zhuang, and Jia Liu. Using statistical analysis and support vector machine classification to detect complicated attacks. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 5:2747–2752, Aug. 2004.
- [120] Dean Turner, Marc Fossi, Eric Johnson, Trevor Mack, Joseph Blackbird, Stephen Entwisle, Mo King Low, David McKinney, and Candid Wueest. Symantec global internet security threat report. White Paper, Symantec Enterprise Security, April 2008. 1
- [121] W. Tylman. Anomaly-based intrusion detection using bayesian networks. *Dependability of Computer Systems, 2008. DepCos-RELCOMEX '08. Third International Conference on*, pages 211–218, June 2008.
- [122] A. Volynkin and V. Skormin. Large-scale reconfigurable virtual testbed for information security experiments. *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*, pages 1–9, May 2007.
- [123] Yao-Guang Wei, De-Ling Zheng, and Ying Wang. Research of a negative selection algorithm and its application in anomaly detection. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 5:2910–2913, Aug. 2004.
- [124] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. Tmix: a tool for generating realistic tcp application workloads in ns-2. *SIGCOMM Comput. Commun. Rev.*, 36(3):65–76, 2006.
- [125] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270, 2002. 7
- [126] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 255–270, New York, NY, USA, 2002. ACM. 11



- [127] Walter Willinger, Vern Paxson, and Murad S. Taqqu. Self-similarity and heavy tails: structural modeling of network traffic. pages 27–53, 1998.
- [128] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Trans. Netw.*, 5(1):71–86, 1997.
- [129] Ji-Qing Xian, Feng-Hua Lang, and Xian-Lun Tang. A novel intrusion detection method based on clonal selection clustering algorithm. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, pages 3905–3910, Aug. 2005. 10
- [130] Hong-Yu Yang, Li-Xia Xie, and Feng Xie. A novel anomaly detection approach based on data field. *International Conference on Machine Learning and Cybernetics, 2008.*, 2:1105–1110, July 2008.
- [131] Wu Yang, Xiao-Chun Yun, and Le-Jun Zhang. Using incremental learning method for adaptive network intrusion detection. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 7:3932–3936, Aug. 2005. 10
- [132] Y. Yasami, M. Farahmand, and V. Zargari. An arp-based anomaly detection algorithm using hidden markov model in enterprise networks. *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, pages 69–69, Aug. 2007.
- [133] Zhenwei Yu and Jeffrey J. P. Tsai. A framework of machine learning based intrusion detection for wireless sensor networks. In *SUTC '08: Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)*, pages 272–279, Washington, DC, USA, 2008. IEEE Computer Society.
- [134] Chunlin Zhang, Ju Jiang, and Mohamed Kamel. Intrusion detection using hierarchical neural networks. *Pattern Recogn. Lett.*, 26(6):779–791, 2005. 10
- [135] Like Zhang and G.B. White. Anomaly detection for application level network attacks using payload keywords. *Computational Intelligence in Security and Defense Applications, 2007. CISDA 2007. IEEE Symposium on*, pages 178–185, April 2007.
- [136] Xue-Qin Zhang and Chun-Hua Gu. Ch-svm based network anomaly detection. *Machine Learning and Cybernetics, 2007 International Conference on*, 6:3261–3266, Aug. 2007.

- [137] Ya-Jing Zhang, Chao-Zhen Hou, Fang Wang, and Li-Min Su. A niching negative selective genetic algorithm for self-nonsel self discrimination in a computer. *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, 1:276–280, 2002.
- [138] Ya-Jing Zhang and Hui-Wen Leng. An investigation of immune detection algorithm with vaccine operator and fuzzy match. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 7:3943–3948, Aug. 2005.
- [139] Ya-Jing Zhang, Yang Xue, and Shuan-Hu Wu. A gene immune detection algorithm with a strategy of dna pri. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 4:2421–2426, Aug. 2004.
- [140] Yu-Fang Zhang, Zhong-Yang Xiong, and Xiu-Qiong Wang. Distributed intrusion detection based on clustering. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 4:2379–2383, Aug. 2005. 10
- [141] Jun-Zhong Zhao and Hou-Kuan Huang. An intrusion detection system based on data mining and immune principles. *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, 1:524–528, 2002.
- [142] Jiang Zhong, Zhiguo Li, Yong Feng, and Cunxiao Ye. Intrusion detection based on adaptive rbf neural network. *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, 2:1081–1084, Oct. 2006. 10
- [143] Yong Zhong, Xiao-Lin Qin, and Dong-Mei Lin. An intrusion detection method based on clustering multidimensional sets. *Machine Learning and Cybernetics, 2006 International Conference on*, pages 2799–2804, Aug. 2006.
- [144] Hong-Gang Zhou and Chun-De Yang. Using immune algorithm to optimize anomaly detection based on svm. *Machine Learning and Cybernetics, 2006 International Conference on*, pages 4257–4261, Aug. 2006. 10
- [145] Urko Zurutuza, Roberto Uribeetxeberria, and Diego Zamboni. A data mining approach for analysis of worm activity through automatic signature generation. In *AISec '08: Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 61–70, New York, NY, USA, 2008. ACM.