IJICTR
International Journal of Information & Communication Technology Research

Volume 12- Number 1 – Winter 2020 (20 -31)

Simulating Benchmark Datasets for Worm Propagation Studies

Sara Asgari
Department of Computer Engineering
Amirkabir University of Technology
Tehran, Iran
s.asgari@aut.ac.ir

Babak Sadeghiyan*
Department of Computer Engineering
Amirkabir University of Technology
Tehran, Iran
basadegh@aut.ac.ir

Received: 30 September 2019 - Accepted: 20 December 2019

Abstract—**Identifying the roots of a worm and reconstructing its spread path are among essential concerns in digital forensics. This knowledge assist the prosecutor in understanding how the attack happened in the network and how security protections were breached. Evaluating methods proposed for this purpose is problematic due to the lack of suitable datasets containing both worm traffic and normal traffic. In this paper, we investigate various approaches of generating such datasets and propose a technique to generate suitable datasets for these evaluations. ReaSE is a tool for creating realistic simulation environments, which considers three aspects, i.e., topology generation, normal traffic generation, and attack traffic generation. We modify ReaSE to make it suitable for generating these datasets. We also generate various datasets for Code Red I, Code Red II, SQL Slammer and modified version of them in different scenarios and make them accessible to the public.**

Keywords-simulation; dataset generation; spread path reconstruction; source detection; worm; Code-Red; SQL Slammer

I. INTRODUCTION

Locating the sources of a worm and reconstructing its spread path are among essential concerns in digital forensics. This information would help the forensic investigator guess which nodes were responsible for entering infection into the network and how security protections were breached. This knowledge is helpful in identifying network security weaknesses and better planning to mitigate future attacks. Despite the fact that the research community has realized this necessity, relatively few methods have been proposed for this purpose to date since evaluating these methods is problematic. Both worm traffic and normal traffic are needed to evaluate these methods. Currently, researchers inject worm traffic into separately-generated normal traffic; for example, in [1] and [2], this approach is used. In this approach, the network race condition (in real networks, worm traffic and normal traffic race each other to achieve bandwidth) is not considered since these two types of traffic are not generated in the same network and concurrently.

Furthermore, obtaining normal traffic is a big challenge since network administrators often avoid exposing their network traffic due to privacy issues. Moreover, available datasets are usually anonymized due to privacy issues, e.g., Kent[3][4], MAWI[5], UGR-16[6], SANTA[7], UNIBS[8], PUF[9], LBNL[10]. Currently, there are no suitable datasets for evaluating worm source and spread path identification methods, i.e., the datasets that include both worm traffic and normal traffic generated concurrently and in the

* Corresponding Author

same network. If such datasets existed, the network race condition would be considered. In addition, the main concern of researchers for evaluating worm source and spread path identification methods, which is collecting normal traffic, would be addressed.

Our purpose is to suggest a technique for generating suitable datasets for worm propagation studies, as well as to generate a number of these datasets. To do this, we examine different approaches of generating datasets for worm propagation studies. We study 21 datasets and various traffic generators to obtain normal traffic. We also investigate various technologies of creating worm experimental environments. The result of our investigations is that ReaSE[11] is an appropriate tool for generating the datasets described above. However, some changes need to be made to it. So we modify ReaSE and also generate multiple datasets for SQL Slammer[12], Code-Red I, Code-Red II[13], and modified versions of them in various scenarios.

The rest of this paper is structured as follows: Section II introduces the related works of generating datasets. Section III discusses possible dataset generation approaches for evaluating worm source and spread path identification methods. Section IV describes our proposed technique. This section provides a short description of ReaSE and its features, as well as the changes we made to ReaSE. Section V details our generated datasets. Finally, Section VI concludes the paper and gives an outlook to future work.

II. Related Works

So far, many different efforts have been made to generate suitable datasets for evaluations, e.g., IDS evaluations.

One of the most popular datasets for intrusion detection is DARPA dataset[14][15][16], which was generated in 1998/1999 and includes different kinds of attacks, e.g., port scan, buffer overflow, and DoS. In [17], [18] and [19], DARPA dataset is criticized. These criticisms show that DARPA dataset does not illustrate the realistic behavior of network traffic. KDD Cup[20] is based on the DARPA dataset and contains various kinds of attacks, e.g., DoS. The analysis performed in [21] shows that there are important issues with KDD Cup that lead to very poor evaluations. One of the most important drawbacks of KDD Cup is the large number of duplicates[21]. NSL-KDD[22] improved KDD Cup and removed redundancies. These datasets have been generated many years ago and are outdated.

Kyoto 2006+[23] dataset was collected from diverse types of honeypots in 2011. Although it includes various attacks, e.g., port scans, DoS, shellcode, and malware, it contains a small amount of normal traffic. ISCX[24] dataset contains both normal traffic and some kinds of attack traffic, e.g., DoS, DDoS, and SSH brute force. It was generated in 2012 by capturing traffic in an emulated network environment. In order to generate this dataset, two profiles were considered. α-profile defines a description of an attack scenario, and β-profile characterizes normal user behavior. ISCX dataset does not include HTTPS traffic. However, nowadays, nearly 70% of network traffic is HTTPS[25]. Moreover, the testbed of generating ISCX dataset contains very few nodes. TUIDS[26][27] dataset was generated within an emulated environment in 2012 and contains normal traffic and some primary attacks, e.g., Dos, DDoS, and port scanning. TUIDS includes three parts: a TUIDS intrusion dataset, a TUIDS coordinated scan dataset, and a TUIDS DDoS dataset. SANTA dataset[7] is a flow-based dataset captured within an ISP environment in 2014 and contains real network traffic. IRSC[28] was generated in 2015 by capturing normal and attack traffic from the internet and also running manual attacks. SANTA and IRSC are not accessible to the public for privacy reasons. Kent dataset[3][4] was generated in 2016 by capturing the traffic of the Los Alamos National Laboratory network within 58 days. This dataset is heavily anonymized, e.g., IP, time, and port, due to privacy issues. CICIDS dataset[25] was generated within an emulated environment in 2017. For generating a wide range of attack types, six attack profiles were created, i.e., heartbleed, DDoS, DoS, botnet, brute force, web, and infiltration. Moreover, the abstract behavior of human interactions was profiled by the B-profile system and used to generate normal traffic. The testbed of generating the CICIDS dataset is very small. PUF[9] is a DNS dataset captured within a campus network in 2018, and is available in the flow-based format. Due to privacy issues, IP addresses are removed from this dataset.

None of the mentioned datasets are suitable for evaluating worm source and spread path identification methods. To the best of our knowledge, there is no suitable dataset containing both worm traffic and normal traffic for evaluating these methods. Due to the lack of such datasets, researchers usually inject the worm spread traffic to separately-generated normal traffic. In this approach, worm spread traffic is usually generated using simulation. Then this traffic is combined with normal traffic obtained from real networks or available datasets. For example, in [29], the worm spread traffic simulated with GTNetS[30] was combined with normal traffic of the ISCX dataset[24]. Also, In [1], traffic of a real network was recorded and used as normal traffic.

In this paper, we propose a technique to generate suitable datasets for worm propagation studies and also generate a number of these datasets, which enable researchers to evaluate their proposed methods without facing any challenge, e.g., [31].

III. Dataset Generation Approaches

To evaluate methods proposed for detecting the source of a worm and reconstructing its spread path, we require datasets that include both worm traffic and normal traffic. There are two approaches for creating such datasets:

1) injecting separately-generated worm traffic into normal traffic

2) Generating both worm traffic and normal traffic concurrently.

Furthermore, our studies show that there are three methods for obtaining worm traffic and normal traffic: using available datasets, collecting traffic of real networks, traffic generation in the experimental environment. We investigate these methods in this section.

B. Using Available Datasets

To obtain normal traffic for evaluating worm source and spread path identification methods, we investigate 21 datasets. The results of our investigations show that available datasets have some issues to be used for these evaluations. We categorize these issues as below:

- Some of the datasets were created many years ago, e.g., DARPA, KDD CUP, NSL-KDD, and PU-IDS, date back to 1998/1999. Networks and their traffic have been changed a lot since then, e.g., new protocols have been introduced, and some protocols have been outdated. Therefore, these datasets are deprecated and should not be used for today's evaluations.
- The format of available datasets can be categorized as below[32]:
  - Flow-based: In the flow-based format, all packets with a number of same properties within a time window are aggregated into a single flow. Flow-based data only contain metadata about network connections.
  - Packet-based: Packet-based data are usually captured in pcap format.
  - Other: Some datasets have no standard format.

  To evaluate worm source and spread path identification methods, traffic should be available in packet-based format because in flow-based or other formats, some essential data for evaluations are not provided. Note that packet-based data can be converted to flow-based or other formats, but not vice versa.
- Because of privacy reasons, some datasets are not accessible to the public.
- In some datasets, attack traffic and normal traffic are combined. These datasets do not provide normal traffic separately.
- Some necessary information, e.g., link delay, bandwidth, and network topology, to inject worm traffic into normal traffic is not available in some datasets, such as LBNL.
- Some of the datasets do not contain a number of new network protocols. For example, there are no HTTPS traces in many of the available datasets. However, nowadays, nearly 70% of network traffic is HTTPS[25].
- Some datasets, such as CICIDS[25] and ISCX, have very small testbeds.
- Available datasets are often sanitized, and all of the sensitive information, e.g., IP addresses, payloads, times, and ports, are anonymized or removed from them to be distributed without any privacy issues. Sanitization makes datasets inadequate for evaluating some worm source and spread path identification methods as it may remove some necessary information needed for evaluations.

Some example datasets for each issue are illustrated in Table I.

C. Collecting Traffic of Real Networks

The traffic of real networks contains sensitive information of network users. Therefore, due to privacy issues and potential security threats, network administrators often refuse to expose the traffic of their network, and if they do so, they will anonymize it. Moreover, we cannot ensure that the traffic traces captured in real networks only contain normal traffic, due to the attack traffic which may exist.

TABLE I. SOME EXAMPLE DATASETS FOR EACH ISSUE

Issue	Example Datasets
Outdated	PU-IDS[33], NSL-KDD[22], KDD Cup[20], DARPA[14][15][16]
Not including new protocols	PU-IDS[33], NSL-KDD[22], KDD Cup[20], ISCX[24], DARPA[14][15][16]
Being sanitized	IP: LBNL[10], Kent[3][4], UGR 16[6], MAWI[5], UNIBS[8], PUF[9], Unified Host and Network[34]
	Payload: LBNL[10], SANTA[7], MAWI[5], CTU-13[35]
	Time: Kent[3][4], Unified Host and Network[34]
	Ports: Kent[3][4]
Not accessible to the public	SANTA[7] and IRSC[28]
Not available in packet-based format	KDD Cup[20], Kent[3][4], NSL-KDD[22], PU-IDS[33], PUF[9], Unified Host and Network[34], UNIBS[8], SANTA[7], SSENET-2014[36], SSENET-2011[37] and UGR 16[6]
Not providing normal traffic separately	CTU-13[35], MAWI[5], NGIDS-DS[38], TUIDS[26][27] and UNSW-NB15[39]
Lack of access to necessary information for traffic injection	LBNL[10], MAWI[5], PUF[9], Unified Host and Network[34], UGR 16[6], Kent[3][4], SANTA[7], CTU-13[35], UNIBS[8]
Small testbed	CICIDS[25], ISCX[24]

D. Traffic Generation in Experimental Environment

1) Normal Traffic: the traffic generated by most traffic generators is uni-directional. Some traffic generators generate bi-directional traffic, but there is no request-response interaction between two sides of communication. The traffic generated by these two categories of traffic generators is not suitable for evaluating worm propagation studies. However, it can be helpful in other applications, e.g., performance evaluation, quality of service measurement, and optimization. The normal traffic used to evaluate worm propagation studies should be bi-directional traffic that each side of communication affects the other side, e.g., request-response communications in application-layer protocols and retransmitting packets in TCP.
Below, we provide a brief discussion on some traffic generators:

- Swing[40][41]: Swing captures the packet interactions of applications and extracts distributions for application, user, and network behavior. Then, in an emulated environment, it generates traffic based on the underlying model. One of the limitations of swing is that it generates realistic traces for a single link.
- Tmix: Tmix is a traffic generation system described in [42] for the NS-2 simulator[43]. Tmix takes a packet header trace obtained from a network link as input and reverse-compiled it to produce a source-level characterization of each TCP connection in the trace. Then it uses this characterization to emulate the socket-level behavior of the source application that created the connections in the trace. Tmix is also implemented for GTNetS[30] in [44]. Tmix, like swing, generates traffic for a single link.
- PackMime-HTTP: PackMime-HTTP is a model and implementation proposed in [45] for generating realistic synthetic web traffic in NS-2 simulator. The source variable generation model of PackMime-HTTP was extended and modified in [46] to work with the NS-3 simulator [47]. Moreover, an additional working mode was added to it. These traffic generators are only able to generate HTTP traffic.
- Ammar et al.[48] developed a tool for generating traffic for the NS-3 simulator based on the PPBP (Poisson Pareto Burst Process) model [49]. The traffic generated by this traffic generator is uni-directional.
- NeSSi [50]: NeSSi is a network simulation tool that provides various network security capabilities. NeSSi provides a scalable and distributed architecture built on top of the JIAC[51] framework. NeSSi includes a small number of application-layer protocols.

2) Worm Traffic: The technologies of creating a worm experimental environment are classified as follows in [52]: hardware testbed, network emulation, packet-level simulation, analytical model, and hybrid method. The fidelity of analytical models, as discussed in [52], is inadequate for our purpose. The use of network emulation would not fulfill the required scalability for worm propagation researches. Furthermore, since worm experiments involve using a lot of nodes, providing hardware testbeds is not feasible. Packet-level simulation has better fidelity than the analytical model. It also provides sufficient scalability. Following is a brief discussion of several network simulators that provide packet-level worm simulation:

- SSFNET: Liljenstam et al.[53] developed a model for large-scale worm attacks and made it available as an add-on package to the SSFNET simulator [54][55]. They use the detailed packet-level simulation for part of the network, and a less accurate but computationally efficient model for other parts[56].
- NS-2: NS-2 includes some behavioral models of worms. NS-2 uses the same approach we mentioned above for SSFNet. Furthermore, scanning worm models are problematic since there is no mechanism for assigning IP addresses to nodes in NS-2 [56].
- GTNetS: various worms can be simulated in GTNetS by adjusting a number of parameters, e.g., number of simultaneous connections, transport layer protocols, scan rate, infection length.
- PAWS [57]: PAWS is a distributed worm propagation simulator. In PAWS, to improve the simulation speed, worm propagation behavior is simplified. For example, only those scans are delivered to the destination that the destination host has not yet infected. To decrease overhead, PAWS also aggregates the packets intended for transmission to a node into a single message and transfers that message at the end of the time unit.
- A worm propagation simulator is implemented in the Perl language in [58]. While this simulator takes into account the effect of propagation delays and link bandwidth, it disregards competing traffic, loss, and queuing. Furthermore, TCP-based worms use a simplified TCP model, in which some features, e.g., congestion window and slow start, are not considered [56].
- NeSSi: In addition to generating normal traffic, we can also simulate worm propagation in Nessi. The Blaster and SQL SQL Slammer worms can be simulated using NeSSi's worm propagation scheme. The worm model provided by NeSSi can be

extended. Furthermore, researchers are able to develop a new worm model.

IV. Our Proposed Technique

According to our investigations, ReaSE can be an appropriate tool to generate suitable datasets for evaluating worm source and spread path identification methods, but some necessary changes are needed. In this section, we will first provide an overview of ReaSE and its capabilities, followed by an explanation of the changes we made to it.

A. OMNeT++ Overview

OMNeT++[59] is a component-based, modular C++ simulation library and framework, primarily for building network simulators. OMNeT++ model consists of hierarchical modules communicating with each other by message passing. This model is illustrated in Fig. 1. At the lowest level of this hierarchy are simple modules that realize the functionalities and combine one or more C++ classes. One or more simple modules form a compound module. Compound modules can be connected to each other using channels, i.e., outgoing and incoming gates. Each channel has a certain delay and bandwidth. A compound module can realize a complete functionality, e.g., the functionality of a router. The top-level module in this hierarchy is the system module. The system module is made up of simple and compound modules.

For simulating with OMNeT++, two types of files are necessary:

1) NED files: NED language is used to specify the topology of a model. Each NED file, which has the .ned suffix, facilitates the modular description of a network, and contains import directives and the definitions of channels, network, and simple and compound modules.

2) Omnetpp.ini file: Omnetpp.ini is the configuration file that describes the Configuration and input data for the simulation.

As an example, consider the scenario in Fig. 2. This network consists of three nodes, one router, and two hosts. Omnetpp.ini specifies that the starting module is MyNetwork. This compound module, defined in MyNetwork.ned file, consists of several submodules (clients and router) and the interconnections between them. The functionality of the router and host is realized with the compound modules Router and StandardHost, respectively. Moreover, each of these modules themselves consists of several submodules. For example, the Router consists of modules NetworkLayer and RoutingTable.

The INET[60] is an open-source OMNeT++ model suite for wired, wireless, and mobile networks. Common internet protocols, e.g., ICMP, TCP, UDP, and IP, as well as intermediate and end systems, e.g., hosts and routers, can be simulated using INET.

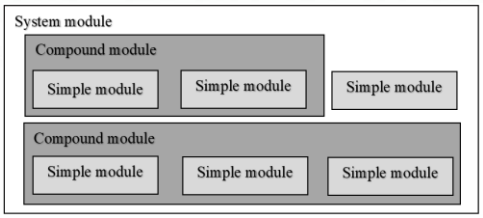

Figure 1. OMNeT++ model.

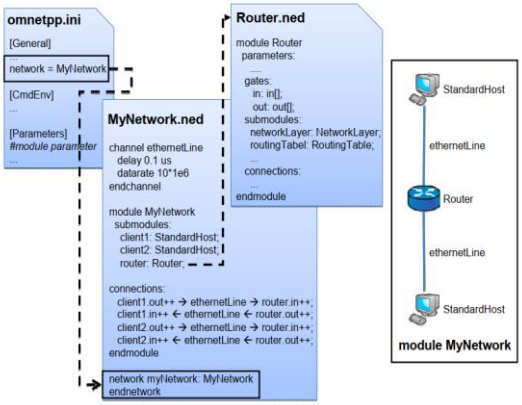

Figure 2. An example of simulation setup[11].

B. ReaSE Overview

ReaSE is a tool developed upon the INET framework and creates a realistic simulation environment by considering three aspects, i.e., topology generation, normal traffic generation, and attack traffic generation. This tool is accessible to the public on https://i72projekte.tm.uka.de/trac/ReaSE.

1) Topology generation: Topology generation in ReaSE is made up of two parts. First, the connections of Autonomous Systems (AS) are created. Autonomous Systems are classified into two categories: stub and transit. The topology inside each AS is then generated. As illustrated in Fig. 3., the structure of each AS is hierarchical, including three layers edge, gateway, and core. ReaSE uses PFP (positive-feedback preference) model[61] to generate realistic topologies. The PFP model randomly generates topologies that show the power-law distribution in node degrees. The rich club feature[62] is also considered. Furthermore, ReaSE uses the heuristically optimal topology (HOT) approach[63] for topology generation within an AS. In the HOT approach, for generating topology inside each AS, in addition to power-law distribution, market demands, link costs, and hardware constraints are also considered. This approach leads to a hierarchical topology that the number of nodes and connectivity increase from the core to edge, whereas link bandwidth decreases.

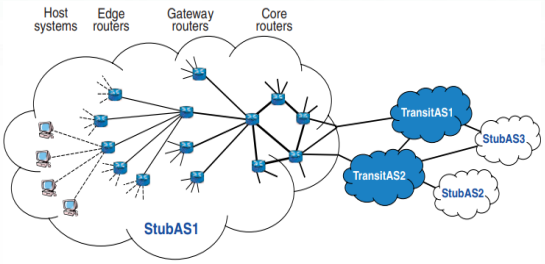

Figure 3. Hierarchical topologies generated by ReaSE[64].

2) Normal traffic generation: ReaSE can generate realistic normal traffic between hosts. Realistic in this case means that the generated traffic exhibits self-similar behavior[65] and is based on a combination of different types of traffic.

ReaSE defines eight traffic profiles, i.e., web, backup, mail, interactive, streaming, ping, nameserver, and Misc. By setting various parameters, i.e., Reply Length, Request Length, Reply Per Request, Requests Per Flow, Time To Respond, Time Between Requests, Time Between Flows, WAN Probability, and Selection Probability, the behavior of each traffic profile can be defined. So, by using these traffic profiles and setting their parameters appropriately, we can generate various patterns of normal traffic. To generate a specific traffic pattern, the settings, i.e., traffic profiles and the parameters, should be given as an XML configuration file to the simulator. As an example, Fig. 4. represents part of such a configuration file.

3) Attack traffic generation: ReaSE enables simulation of DDoS attacks and worm propagations. Although it is mentioned in [11] that both TCP-based and UDP-based worms were implemented in ReaSE, the TCP-based worm is not accessible to the public. However, various UDP-based worms can be simulated by setting parameters, e.g., payload length, the range of IP addresses to scan, infection port, time between probing packets, in the omnetpp.ini file.

```
<Profile>
        <Id><3>
        <Label><Web Traffic>
        <RequestLength><200>
        <RequestsPerSession><10>
        <ReplyLength><1000>
        <ReplyPerRequest><30>
        <TimeBetweenRequests><2.0>
        <TimeToRespond><0.5>
        <TimeBetweenSessions><3.0>
        <Ratio><36.0>
        <WANRatio><73.0>
</Profile>
<Profile>
        <Id><21>
        <Label><Ping>
        <RequestLength><64>
        <RequestsPerSession><3>
        <ReplyLength><64>
        <ReplyPerRequest><4>
        <TimeBetweenRequests><1.0>
        <TimeToRespond><0.2>
        <TimeBetweenSessions><1.0>
        <Ratio><4.3>
        <WANRatio><50.0>
        <Hoplimit><255>
</Profile>
```

Figure 4. An example file: traffic profiles and their parameters

C. *ReaSE Modifications*

Below, we shortly justify the changes we made to ReaSE and do not go through the technical aspects:

- Nodes are divided into two categories in ReaSE: nodes that can only generate normal traffic and nodes capable of only generating worm traffic. For evaluating worm propagation studies, worm traffic and normal traffic should be generated by the same nodes. So we add the module that implements worm propagation functionality (e.g., udpWormVictim.ned in UDP-based worms) in the nodes that generate normal traffic and remove the nodes that only generate worm traffic from the simulator.
- In ReaSE, servers only listen to a specific port and respond to requests. However, they cannot send requests to other servers, unlike real servers. So we add this capability to servers. To accomplish this, we add the InetUser module of ReaSE to the servers and make required changes to InetUser and ConnectionManager modules, as well as the InetUser class. So each server can also communicate with other servers.
- To generate datasets in packet-based format, each node's traffic should be captured in pcap format. We provide this capability by adding the TCPDump module of INET to nodes.
- Although in real networks, hosts use random source ports for communications, In ReaSE, each traffic profile uses a fixed source port to generate traffic. We consider this randomness in source port selection by modifying the TrafficProfile struct of ReaSE.
- In ReaSE, each server does not necessarily listen to requests on a unique port, unlike real servers. We set the port parameter of each server to listen on a unique port.
- To generate normal traffic containing new network protocols, we add four traffic profiles, i.e., FTP, SSH, HTTP, and HTTPS, to ReaSE and create a server for each of them (except SSH). To accomplish this, we create a compound module for each server, i.e., FTPServer.ned, HTTPSServer.ned and HTTPServer.ned.
- The TCP-based worm implemented in ReaSE is not accessible to the public. TCP-based worms establish a TCP connection to each target machine before sending payload packets. Furthermore, they use multiple concurrent TCP connections to infect multiple machines at the same time, speeding up the propagation process. Thus, by considering the behaviors of TCP-based worms, we implement a model that various TCP-based worms with different scanning strategies, such as uniform random scanning and local preference scanning, and propagation models SIR (Susceptible-Infected-Recovered) or SI (Susceptible-Infected) can be simulated by

setting different parameters, e.g., infection port, infection length, the range of IP address to scan, the number of simultaneous connections, preference probability, and recovery probability. We implement two classes, TCPWorm and TCPWormThread. TCPWorm class is responsible for storing threads in a list and managing them. TCPWormThread class handles each thread. We also implement a module named TCPWorm, which consists of these two classes, and add it to vulnerable hosts.

- The model implemented in ReaSE for UDP-based worms is uniform random scanning and SI. We change this model such that both uniform random scanning and local preference scanning worms with propagation models SI or SIR can be simulated by setting the parameters preference probability and recovery probability. Furthermore, the UDP-based worm in ReaSE always sends probing packets from a fixed source port. We modify it to scan IP addresses using random source ports.

D. *Generating Datasets*

Various datasets containing both worm traffic and normal traffic can be generated using the technique proposed in this section. Using different traffic profiles and adjusting their parameters, Various patterns of normal traffic can be generated. Various random network topologies can also be generated by setting a number of parameters. Furthermore, we can simulate various types of local preference and uniform random scanning worms by simply adjusting different parameters. Other kinds of scanning worms, such as sequential scanning worms, can also be simulated by making small changes.

E. *Validation*

Multiple experiments were carried out in [64] to validate that the normal traffic and topologies generated by ReaSE have realistic characteristics. These experiments yielded the following results:

- Although there are some deviations in a small number of nodes, the topologies generated by ReaSE display the power-law distribution in node degrees well.
- The generated background traffic shows self-similar behavior.

It is worth noting that we made no changes to ReaSE's topology generation capability. Furthermore, ReaSE employs two mechanisms to achieve self-similar traffic behavior:

- Using several traffic sources that are switched on and off based on heavy-tailed intervals.
- Using heavy-tailed packet sizes for different traffic flows.

We did not interfere in the operation of these mechanisms. So the self-similarity of normal traffic has been preserved.

V. Our Generated Datasets

We generate two categories of datasets, each contains several sets of traffic traces[66]. In this section, we describe how we generate these datasets. We will make these datasets accessible to the public on https://github.com/Sara-Asgari/Datasets.

In order to create realistic topologies, one approach is taking advantage of real-world network observations. The topologies created using this approach are very realistic. Another approach is random topology generation[67], which is widely used in the research community[11]. We generate two categories of datasets; each uses one of these approaches.

A. *Category I*

The topology and link parameters of the network in which the category I datasets were generated are illustrated in Fig. 5 and Table II, respectively. This network was derived from a simplified version of a large ISP in Italy, described in [68]. To design this network, we also consider network properties in [69] and the common topology of today's networks. For ensuring network availability, this network provides redundancies, too. It consists of four subnets and is made up of four core nodes, eight gateway nodes, sixteen edge nodes, and two hundred end-hosts.

To generate normal traffic, we extracted the type and percentage of application-layer protocols from the CICIDS dataset's first-day traffic (shown in Table III) and used these values to choose the traffic profiles and their selection probabilities. We also extracted the approximate values of the parameters (on average) for traffic profiles from this dataset.

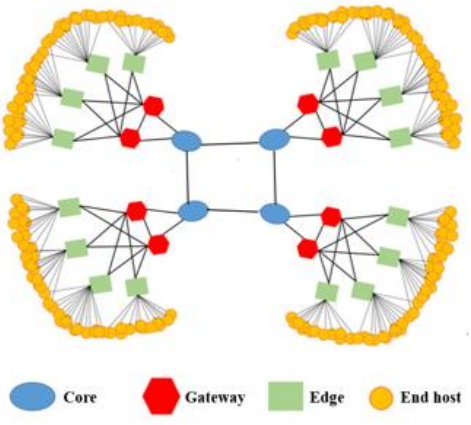

Figure 5. Network topology - category I.

TABLE II. BANDWIDTHS AND DELAYS OF LINKS - CATEGORY I

Link	Bandwidth	Delay
Core to Core	50 Gbps	3 ms
Core to Gateway	20 Gbps	2 ms
Gateway to Edge	10 Gbps	0.25 ms
Edge to Server	2.5 Gbps	5 µs
Edge to Client	100 Mbps	5 µs

TABLE III. TRAFFIC PROFILES - CATEGORY I

Traffic Profile	%
HTTP	53.85%
HTTPS	38.13%
DNS	6.87%
SSH	0.78%
FTP	0.20%
Email	0.14%
Ping	0.03%

TABLE IV. OUR GENERATED DATASETS

	dataset	Worm parameters							Infection network parameters	
		Name	*Transport layer protocol*	*The number of threads (TCP worms)*	*Time between probing packets (UDP worms)*	*Scanning strategy*	*Preference probability (local preference scanning worms)*	*Recovery Probability*	*Number of nodes*	*Type of nodes*
Category I	Set 1	SQL Slammer	UDP	-	Uniform(4ms,8ms)	Uniform Random	-	10^{-4} per ms	30	Client, HTTPS Server and HTTP Server
	Set 2	Modified Slammer	UDP	-	Uniform(5ms,10ms)	Local Preference	$\frac{1}{8}$: random $\frac{1}{2}$: same class A $\frac{3}{8}$: same class B	10^{-4} per ms	28	HTTP Server
	Set 3	Modified Slammer	UDP	-	Uniform(5ms,10ms)	Local Preference	0.3: random 0.7: same subnet	10^{-4} per ms	35	Client
	Set 4	Code-Red I	TCP	23	-	Uniform Random	-	10^{-4} per ms	28	HTTP Server
	Set 5	Code-Red II	TCP	25	-	Local Preference	$\frac{1}{8}$: random $\frac{1}{2}$: same class A $\frac{3}{8}$: same class B	10^{-4} per ms	28	HTTP Server
	Set 6	Modified Code-Red II	TCP	25	-	Local Preference	0.3: random 0.7: same subnet	10^{-4} per ms	35	Client
Category II	Set 1	Modified Code-Red II	TCP	20	-	Local Preference	0.3: random 0.7: same subnet	10^{-5} per ms	52	HTTP Server
	Set 2	Modified Slammer	UDP	-	Uniform(10ms,12ms)	Local Preference	0.3: random 0.7: same subnet	10^{-5} per ms	52	HTTP Server

We generate six sets, each contains three traffic traces with different worm sources and spread path, in Category I (Table IV).

Code-Red II was a TCP-based, local preference scanning worm that exploited a buffer overflow vulnerability in IIS web servers. Since its scanning technique was the local preference, it scanned local IP addresses with more probability than others. More precisely, $\frac{1}{8}$ of the time, it sent probing packets to random IP addresses on port 80, looking for other hosts to infect, $\frac{1}{2}$ of the time, it sent probing packets within the same class A range of the local IP addresses, and $\frac{3}{8}$ of the time, it scanned the same class B range of the local IP addresses. The number of threads spawned by an infected host was 300 for non-Chinese systems and 600 for Chinese systems. Code-Red I was a TCP-based, uniform random scanning worm, i.e., it found vulnerable hosts by sending probing packets to random IP addresses[70]. Similar to Code-Red II, it exploited a security hole in IIS web

servers. SQL Slammer, also known as Sapphire, was a UDP-based, uniform random scanning worm with a total size of 376 bytes which exploited a buffer overflow vulnerability in Microsoft's SQL servers and Microsoft SQL Server Desktop Engines by sending a single UDP packet to port 1434. It was the fastest worm in history, with the maximum scanning rate of 55 million scans per second[71].

The simulated worms in Set 1, Set 4, and Set 5 are similar to the original versions of SQL Slammer, Code-Red I, and Code-Red II, respectively, but their scanning rates are much lower than the original versions. This is because of the fact that original versions spread in the internet containing several thousands of machines, while our simulated network contains much lower nodes, and the large scanning rate causes the network to become infected in a split second, which is not suitable for evaluations. One of our goals is dataset generation for different worms in different scenarios. So we also change some parameters of Code-Red II and SQL Slammer and simulate modified versions of them in Set 2, Set 3 and Set 6.

Similar to real networks, in our simulated networks, only some hosts have the specific vulnerability. The number and type of these nodes are illustrated in Table IV. Other nodes are not vulnerable and only play the role in generating normal background traffic.

In our simulations, we concentrate on spreading part of worms while ignoring their attacking part.

B. Category II

In category II, we generate the network topology using ReaSE's topology generation capability. This topology and the link parameters are illustrated in Fig. 6 and Table V, respectively. This network comprises ten core nodes, twenty gateway nodes, 152 edge nodes, and 1162 end-hosts, located in ten subnets.

Table VI shows the percentage and type of application-layer protocols used to generate normal traffic. Furthermore, the values assigned to traffic profile parameters vary from those assigned to category I.

We generated two sets of traffic traces in category II. The parameters are illustrated in Table IV.

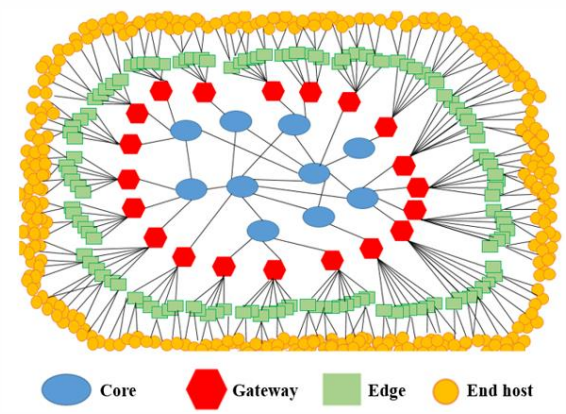

Figure 6. Network topology -category II.

TABLE V. BANDWIDTHS AND DELAYS OF LINKS - CATEGORY II

Link	Bandwidth	Delay
Core to Core	40 Gbps	4 ms
Core to Gateway	16 Gbps	2.5 ms
Gateway to Edge	8 Gbps	0.3 ms
Edge to Server	2 Gbps	10 µs
Edge to Client	80 Mbps	10 µs

TABLE VI. TRAFFIC PROFILES - CATEGORY II

Traffic Profile	%
HTTPS	49.2%
HTTP	35.5%
DNS	8.9%
FTP	3.3%
Email	2.8%

VI. CONCLUSION AND FUTURE WORK

Currently, suitable datasets, including worm traffic and normal traffic, for evaluating worm source and spread path identification approaches do not exist. Therefore, researchers face many difficulties in evaluating their proposed methods. In this paper, we addressed this problem and introduced a technique to generate these datasets. ReaSE is a tool developed on top of the INET framework, an extension of OMNeT++, and creates a realistic simulation environment for IP-based networks by considering three aspects, i.e., random topology generation, normal traffic generation, and attack traffic generation. In this paper, we modified ReaSE to get suitable for generating the datasets for evaluating worm propagation studies. We also validated that the generated topologies and normal traffic demonstrate realistic characteristics. Moreover, we generated eigth sets of traffic traces, each contains three traffic traces in pcap format, for TCP-based and UDP-based scanning worms in different scenarios and will make them accessible to the public soon.

In this paper, we focus on simulating the propagation behavior of worms. Simulating attack behavior when a node becomes infected, e.g., launching DoS attack, is out of the scope of this paper. However, considering both, i.e., propagation behavior and attack behavior, will lead to more realistic simulations, which we leave for future work. Another future work is generating datasets for large-scale networks.

REFERENCES

[1] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm origin identification using random moonwalks," in 2005 IEEE Symposium on Security and Privacy (S&P'05), pp. 242–256, 2005.

[2] W. Shi, Q. Li, J. Kang, and D. Guo, "Reconstruction of worm propagation path by causality," in 2009 IEEE International Conference on Networking, Architecture, and Storage, pp. 129–132, 2009.

[3] A. D. Kent, "Cyber security data sources for dynamic network research," in Dynamic Networks and Cyber-Security, World Scientific, pp. 37–65, 2016.

[4] A. D. Kent, "Comprehensive, multi-source cyber-security events data set," Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2015.

[5] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in Proceedings of the 6th International Conference, pp. 1–12, 2010.

[6] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs," Computers & Security, vol. 73, pp. 411–424, 2018.

[7] C. Wheelus, T. M. Khoshgoftaar, R. Zuech, and M. M. Najafabadi, "A Session Based Approach for Aggregating Network Traffic Data--The SANTA Dataset," in 2014 IEEE International Conference on Bioinformatics and Bioengineering, pp. 369–378, 2014.

[8] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "Gt: picking up the truth from the ground for internet traffic," ACM SIGCOMM Computer Communication Review, vol. 39, no. 5, pp. 12–18, 2009.

[9] R. Sharma, R. K. Singla, and A. Guleria, "A New Labeled Flow-based DNS Dataset for Anomaly Detection: PUF Dataset," Procedia Computer Science, vol. 132, pp. 1458–1466, 2018.

[10] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A first look at modern enterprise traffic," in Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, p. 2, 2005.

[11] T. Gamer and M. Scharf, "Realistic simulation environments for IP-based networks," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, p. 83, 2008.

[12] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," IEEE Security and Privacy, vol. 1, no. 4, pp. 33–39, 2003.

[13] D. Moore, C. Shannon, and K. Claffy, "Code-Red: a case study on the spread and victims of an Internet worm," in Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, pp. 273–284, 2002.

[14] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," Computer networks, vol. 34, no. 4, pp. 579–595, 2000.

[15] "Datasets." [Online]. Available: https://www.ll.mit.edu/r-d/datasets?keywords=DARPA

[16] R. P. Lippmann et al., "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, vol. 2, pp. 12–26, 2000.

[17] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 4, pp. 262–294, 2000.

[18] M. V Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection," in International Workshop on Recent Advances in Intrusion Detection, pp. 220–237, 2003.

[19] J. McHugh, "The 1998 lincoln laboratory ids evaluation," in International Workshop on Recent Advances in Intrusion Detection, pp. 145–161, 2000.

[20] "KDD Cup 1999 Data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in 2009 IEEE symposium on computational intelligence for security and defense applications, pp. 1–6, 2009.

[22] "NSL-KDD dataset." [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[23] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," in Proceedings of the first workshop on building analysis datasets and gathering experience returns for security, pp. 29–36, 2011.

[24] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," Computers and Security, vol. 31, no. 3, pp. 357–374, 2012.

[25] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in ICISSP, pp. 108–116, 2018.

[26] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards Generating Real-life Datasets for Network Intrusion Detection.," IJ Network Security, vol. 17, no. 6, pp. 683–701, 2015.

[27] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in International Conference on Contemporary Computing, pp. 322–334, 2012.

[28] R. Zuech, T. M. Khoshgoftaar, N. Seliya, M. M. Najafabadi, and C. Kemp, "A new intrusion detection benchmarking system," in The Twenty-Eighth International Flairs Conference, 2015.

[29] T. Tafazzoli and B. Sadeghiyan, "A four-step method for investigating network worm propagation," in 2019 7th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–7, 2019.

[30] G. F. Riley, "The georgia tech network simulator," in Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research, pp. 5–12, 2003.

[31] S. Asgari and B. Sadeghiyan, "Reconstruction of Worm Propagation Path Using a Trace-back Approach," in 2020 10th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 233–238, 2020.

[32] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," Computers & Security, pp.147-167, 2019.

[33] R. Singh, H. Kumar, and R. K. Singla, "A reference dataset for network traffic activity based intrusion detection system," International Journal of Computers Communications & Control, vol. 10, no. 3, pp. 390–402, 2015.

[34] M. J. M. Turcotte, A. D. Kent, and C. Hash, "Unified host and network data set," ArXiv e-prints, vol. 1708, 2017.

[35] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," Computers & Security, vol. 45, pp. 100–123, 2014.

[36] S. Bhattacharya and S. Selvakumar, "SSENet-2014 dataset: A dataset for detection of multiconnection attacks," in 2014 3rd International Conference on Eco-friendly Computing and Communication Systems, pp. 121–126, 2014.

[37] Ar. Vasudevan, E. Harshini, and S. Selvakumar, "SSENet-2011: a network intrusion detection system dataset and its comparison with KDD CUP 99 dataset," in 2011 second asian himalayas international conference on internet (AH-ICI), pp. 1–5, 2011.

[38] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," Journal of Network and Computer Applications, vol. 87, pp. 185–192, 2017.

[39] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in 2015 military communications and information systems conference (MilCIS), pp. 1–6, 2015.

[40] K. V. Vishwanath and A. Vahdat, “Swing: Realistic and responsive network traffic generation,” IEEE/ACM Transactions on Networking, vol. 17, no. 3, pp. 712–725, 2009.

[41] K. V. Vishwanath and A. Vahdat, “Realistic and responsive network traffic generation,” in Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 111–122, 2006.

[42] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, “Tmix: a tool for generating realistic TCP application workloads in ns-2,” ACM SIGCOMM Computer Communication Review, vol. 36, no. 3, pp. 65–76, 2006.

[43] L. Breslau et al., “Advances in network simulation,” Computer (Long. Beach. Calif)., vol. 33, no. 5, pp. 59–67, 2000.

[44] P. Adurthi and M. C. Weigle, “Realistic TCP Traffic Generation in ns-2 and GTNetS.”

[45] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle, “Stochastic models for generating synthetic HTTP source traffic,” in IEEE INFOCOM 2004, vol. 3, pp. 1546–1557, 2004.

[46] Y. Cheng, E. K. Çetinkaya, and J. P. G. Sterbenz, “Transactional traffic generator implementation in ns-3,” in Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, pp. 182–189, 2013.

[47] “The ns-3 network simulator. ” [Online]. Available: https://www.nsnam.org/.

[48] D. Ammar, T. Begin, and I. Guerin-Lassous, “A new tool for generating realistic internet traffic in ns-3,” in Proceedings of the 4th international ICST conference on simulation tools and techniques, pp. 81–83, 2011.

[49] M. Zukerman, T. D. Neame, and R. G. Addie, “Internet traffic modeling and future technology implications,” IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 587–596, 2003.

[50] R. Bye, S. Schmidt, K. Luther, and S. Albayrak, “Application-level simulation for network security,” in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, p. 33, 2008.

[51] S. Fricke, K. Bsufka, J. Keiser, T. Schmidt, R. Sesseler, and S. Albayrak, “Agent-based telematic services and telecom applications,” Communications of the ACM, vol. 44, no. 4, pp. 43–48, 2001.

[52] K. Xiaohui, X. Fei, L. Jin, and Z. Jianbo, “A Large-Scale Network Worm Emulation Experimental Environment,” in 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control, pp. 837–842, 2011.

[53] M. Liljenstam, D. M. Nicol, V. H. Berk, and R. S. Gray, “Simulating realistic network worm traffic for worm warning system design and testing,” in Proceedings of the 2003 ACM workshop on Rapid malcode, pp. 24–33, 2003.

[54] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski, "Towards realistic million-node internet simulations," in International Conference on Parallel and Distributed Processing Techniques and Applications, 1999.

[55] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, “Modeling the global internet,” Computing in Science and Engineering, vol. 1, no. 1, pp. 42–50, 1999.

[56] G. E. Riley, M. L. Sharif, and W. Lee, “Simulating internet worms,” in The IEEE Computer Society’s 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings., pp. 268–274, 2004.

[57] S. Wei and J. Mirkovic, "A realistic simulation of internet-scale events," in Proceedings of the 1st international conference on Performance evaluation methodologies and tools, pp. 28-es, 2006.

[58] A. Wagner, T. Dübendorfer, B. Plattner, and R. Hiestand, “Experiences with worm propagation simulations,” in Proceedings of the 2003 ACM workshop on Rapid Malcode, pp. 34–41, 2003.

[59] A. Varga, “Discrete event simulation system,” in Proceedings of the European Simulation Multiconference (ESM’2001), pp. 1–7, 2001.

[60] “INET Framework.” [Online]. Available: https://inet.omnetpp.org/.

[61] S. Zhou, G. Zhang, G. Zhang, and Z. Zhuge, “Towards a precise and complete internet topology generator,” in 2006 International Conference on Communications, Circuits and Systems, vol. 3, pp. 1830–1834, 2006.

[62] S. Zhou and R. J. Mondragón, “The rich-club phenomenon in the Internet topology,” IEEE Communications Letters, vol. 8, no. 3, pp. 180–182, 2004.

[63] L. Li, D. Alderson, W. Willinger, and J. Doyle, “A first-principles approach to understanding the internet’s router-level topology,” ACM SIGCOMM Computer Communication Review, vol. 34, no. 4, pp. 3–14, 2004.

[64] T. Gamer and C. P. Mayer, “Simulative evaluation of distributed attack detection in large-scale realistic environments,” Simulation, vol. 87, no. 7, pp. 630–647, 2011.

[65] M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: evidence and possible causes,” IEEE/ACM Transactions on Networking, vol. 5, no. 6, pp. 835–846, 1997.

[66] S. Asgari, B. Sadeghiyan, “Towards Generating Benchmark Datasets for Worm Infection Studies,” in 10th International Symposium on Telecommunications (IST), pp. 1-8, 2020.

[67] M. Al-Tamimi, W. El-Hajj, and F. Aloul, “Framework for creating realistic port scanning benchmarks,” in 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1114–1119, 2013.

[68] L. Chiaraviglio, M. Mellia, and F. Neri, “Minimizing ISP network energy cost: Formulation and solutions,” IEEE/ACM Transactions on Networking, vol. 20, no. 2, pp. 463–476, 2011.

[69] W. El–Hajj, M. Al–Tamimi, and F. Aloul, “Real traffic logs creation for testing intrusion detection systems,” Wireless Communications and Mobile Computing, vol. 15, no. 14, pp. 1851–1864, 2015.

[70] “Code Red II Analysis.” [Online]. Available: https://www.giac.org/paper/gcih/247/code-red-ii-analysis/100825.

[71] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, “The spread of the sapphire/slammer worm,” CAIDA, ICSI, Silicon Defense, UC Berkeley EECS and UC San Diego CSE, 2003.

Sara Asgari received her B.Sc. degree in Information Technology Engineering from Isfahan University of Technology, Isfahan, Iran, in 2017, and her M.Sc. degree in Computer Engineering from Amirkabir Unviersity of Technology, Tehran, Iran, in 2020. She is currently a Ph.D. student in Computer Engineering at Sharif Unviersity of Technology, Tehran, Iran. Her research interests include Network Security, Web Application Security, Digital Forensics and Software Security.

Babak Sadeghiyan received his Ph.D. in Computer Science from University College, University of New South Wales, Australia in 1993. Since then, he has joined as a faculty member to the Department of Computer Engineering in Amirkabir University of Technology, Tehran, Iran. His research interests include all aspects of Information Security. So far, he has been an author to 5 books, more than 50 journal papers and more than 200 conference papers. His current research interests include Intrusion Detection Systems, Privacy Issues, Malware Forensics and Vulnerability Analysis.