PReSET: A Toolset for the Evaluation of Network Resilience Strategies

Alberto Schaeffer-Filho*, Andreas Mauthe†, David Hutchison†, Paul Smith‡, Yue Yu§ and Michael Fry§
*Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
Email: alberto@inf.ufrgs.br
†School of Computing and Communications, Lancaster University, United Kingdom
Email: {andreas, dh}@comp.lancs.ac.uk
‡Safety and Security Department, AIT Austrian Institute of Technology, Austria
Email: paul.smith@ait.ac.at
§School of Information Technologies, University of Sydney, Australia
Email: {tinayu, mike}@it.usyd.edu.au

***Abstract*—Computer networks support many of the services that our society relies on. Therefore, ensuring their resilience to faults and challenges, such as attacks, is critical. To do this can require the execution of *resilience strategies* that perform dynamic reconfiguration of networks, including resilience-specific functionality. It is important that resilience strategies are evaluated prior to their execution, for example, to ensure they will not exacerbate an on-going problem. To facilitate this activity, we have developed a toolset that supports the evaluation of resilience strategies that are specified as event-driven policies. The toolset couples the Ponder2 policy-based management framework and the OMNeT++ simulation environment. In this paper, we discuss the network resilience problem and motivate simulation as a suitable way to evaluate resilience strategies. We describe the toolset we have developed, including its architecture and the implementation of a number of resilience mechanisms, and its application to evaluating strategies that detect and mitigate Internet worm behaviour.**

I. Introduction

Computer networks, including the Internet, are now widely considered a critical infrastructure that our society depends on. Hence, it is of paramount importance to ensure they are resilient to various challenges, such as component failures and attacks. Resilience is thus a key property for the management and protection of network infrastructures. *Resilience management* encompasses some elements of the traditional FCAPS (fault, configuration, accounting, performance, and security) functionalities [1]. It requires the on-demand adaptation of network configurations, including specialised resilience functionality, in response to performance degradation, component faults or security threats. Because of the consequences of defining poor management configurations, e.g., further degrading service when the network is under duress, it is necessary to carefully specify and test their performance before resilience strategies are deployed in the network infrastructure.

For a number of reasons, evaluating approaches to resilience management on testbeds is difficult. In general, the development of testbeds is a time consuming and costly activity, often resulting in them being limited in scale. Larger-scale testbeds, such as those created as part of the EU FIRE initiative[1], are used for a broad spectrum of experimentation, and make use of operational networks to interconnect disparate sites – disrupting them with generated attack traffic, e.g., a Distributed Denial of Service (DDoS) attack, is generally not possible. Consequently, we advocate that resilience management strategies should be evaluated off-line using a simulation environment, allowing the identification of optimal configurations against different types of attacks and other challenges. This requires the development of tools that allow the specification of management strategies that can be executed and evaluated in a simulation environment.

This paper presents PReSET (**P**olicy-driven **Re**silience **S**trategy **E**valuation **T**oolset), a toolset for the simulation of policy-driven resilience strategies. It supports the evaluation of strategies that are expressed using the Ponder2 policy specification language [2]. Policies are used to orchestrate the behaviour of resilience mechanisms, e.g., anomaly detection and traffic shaping systems, that are realised as modules in the OMNeT++ network simulator [3]. Coupling these two technologies enables known-good management strategies to be readily implemented in a real-world deployment. The extensibility and modularity of OMNeT++ enables the rapid development of attack behaviours and prototypical resilience mechanisms to detect and mitigate them. The inherent measurement capabilities of the simulator supports the evaluation of management strategies, for example, in terms of key performance indicators. The toolset is an important element in our overall resilience management framework, which has been first presented in [1].

This paper is organised as follows: Section II will present a brief introduction to the topic of network resilience, discuss the main types of network challenges and attacks, and motivate the use of a simulation environment to model and analyse resilience strategies. Section III will discuss the requirements and design of the policy-driven resilience simulator proposed in this paper. Section IV will develop a case study on worm propagation and detection, showing how the resilience simulator can be used for the evaluation of resilience strategies. Section V outlines the relevant related work. Finally, Section VI

This work has been done while Alberto Schaeffer-Filho was at the School of Computing and Communications, Lancaster University.

[1]Future Internet Research & Experimentation. Available online at: http://cordis.europa.eu/fp7/ict/fire/

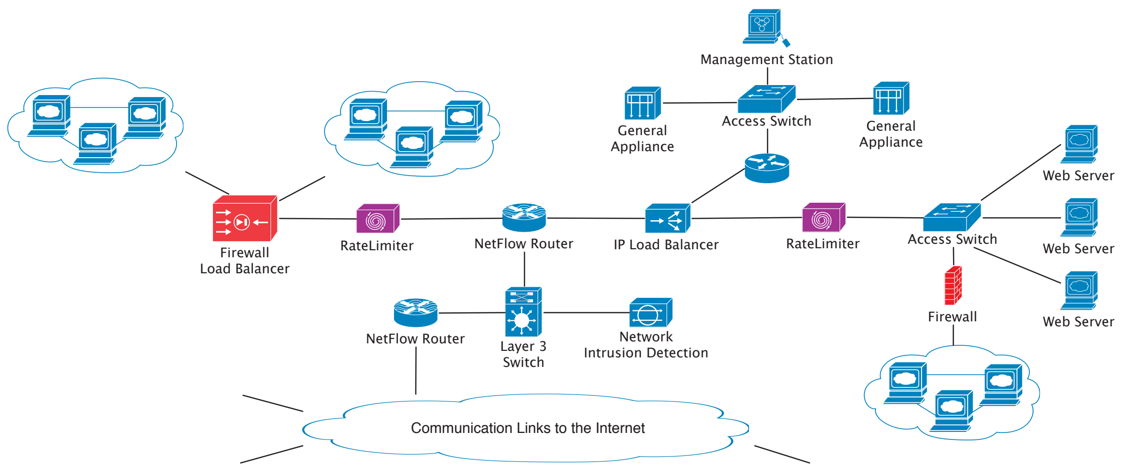


Fig. 1. Network infrastructure with mechanisms implementing a set of resilience functions and services

presents the concluding remarks.

II. Network Resilience

In this section, we characterise challenges and attacks to networks, and present a discussion on the means to analyse the impacts of a challenge and to evaluate resilience strategies.

A. *Network Challenges and Attacks*

Communication networks must be resilient to a multitude of malicious attacks and other challenges to their operation. Sterbenz et al. [4] define resilience as the ability of a network to maintain acceptable levels of operation in the face of faults and challenges. A classification of challenges to the global Internet and interdependent networks has been defined, in which challenges are any characteristic or condition that impacts the normal operation of the network. These may include mis-configuration of equipment, large-scale disasters disrupting the infrastructure, malicious attacks, environmental challenges such as episodic connectivity and weak channels, and unusual but legitimate traffic load [4].

Our research on network resilience has largely focused on malicious behaviour, in particular DDoS attacks and worm propagation. The aim of a DDoS attack is to saturate a target resource, such as a set of network links or servers, with unusually high demands for service [5]. Computer worms, on the other hand, are a type of malware that can quickly propagate in the Internet due to their self-replication capability. Increased network traffic caused by worms can severely disrupt the operation of networks [6]. Although this type of malware has been studied for a number of years, according to recent security reports [7], worms constituted approximately 9% of the successful malware infections in 2011. In previous work, we have presented strategies for detecting and containing DDoS attacks [8]; detection and remediation of worms will be discussed in detail in the form of a case study in Section IV.

A resilience strategy to address the attacks mentioned above requires the management and reconfiguration of interacting detection and remediation mechanisms that operate in the network infrastructure. Initially, *detection mechanisms* support the identification and categorisation of challenges to the network. They may vary from a simple link monitor that can determine whether high volumes of traffic are being observed, to sophisticated detection systems and traffic classifiers that detect anomalous changes in traffic features. Similarly, a range of *remediation mechanisms* may be used for containing the effects of a challenge. For example, various forms of traffic shaping can be used, from simply blocking traffic to probabilistic rate limiting, which can be applied at different protocol levels and to individual network device ports. Firewalls and OpenFlow switches [9], for example, can be used to block or shape network traffic. A typical network infrastructure of the kind we are considering, which includes a range of resilience mechanisms, is shown in Fig. 1.

In our work, resilience strategies are defined as a set of policies that reconfigure the operation of resilience mechanisms at run-time in response to events, such as high link utilisation, malicious attacks or equipment failures, for example [10]. Policy-based management is used to control the operation of these mechanisms in the face of new types of challenges. Thus, different types of mechanisms can be selectively enabled or reconfigured in specific operational contexts.

B. *Cost of Evaluation and Testing*

The evaluation of large-scale challenges, such as DDoS attacks and worm propagations, is difficult because these activities are typically highly distributed in nature and disrupt normal network behaviour. Consequently, resilience strategies to mitigate them can require the coordination of various monitoring and control mechanisms across different administrative domains, protocol levels and heterogeneous infrastructures. The use of testbeds for evaluating network performance and protocol design can involve high costs of hardware and development effort [11]. Moreover, real testbeds are generally not suitable for the evaluation of large-scale challenges that tend to affect multiple autonomous systems.

As an alternative, to mitigate costs and address scaling issues associated with testbeds, we advocate the reproduction of network challenges and resilience mechanisms in a

TABLE I. CAPABILITIES OF THE MOST POPULAR NETWORK SIMULATORS

Evaluation criteria	*NS-2*	*NS-3*	*OPNET*	*OMNeT++*	*SSFNet*	*QualNet*
Platform extensibility	High	High	High	High	High	High
Availability of models	High	Low	High	High	Low	High
Performance & scalability	Low	High	High	High	High	High
Modelling generality	High	High	High	High	High	High

simulation environment [12]. To this end, we have developed a toolset that couples policy-based management and network simulation. Policies specify the required adaptations based on conditions observed during run-time operation of the network (as opposed to hardcoded protocols) [13]. The integrated toolset allows us to analyse a range of wide-scale challenge scenarios and assess the effectiveness of a set of management policies controlling the operation of resilience mechanisms implemented as simulated components.

III. POLICY-DRIVEN RESILIENCE SIMULATOR

The policy-driven resilience simulator presented in this paper is based on an integration between the OMNeT++ simulator [3] and the Ponder2 policy framework [2]. The toolset allows the evaluation of resilience strategies consisting of instrumented mechanisms within the simulation, whose behaviour can be adapted during run-time – e.g., setting flags, dropping connections, triggering or stopping monitoring sessions, etc. The next sections will present the main design decisions and requirements related to this toolset, as well as describe the architecture and the attacks and resilience mechanisms supported.

A. Simulator Requirements

The toolset is based on the integration of a standard network simulator to a policy management framework. We have considered the use of the most popular general purpose network simulators, including NS-2 [14], NS-3 [15], OMNeT++ [16], SSFNet [17], OPNET [18] and QualNet [19]. The choice of a suitable platform was constrained by a number of requirements, namely:

Platform extensibility: the simulator must be extensible, not only in terms of protocols models, but also its ability to be instrumented to allow communication with the policy framework;

Availability of models: the availability of a large number of network models and protocol implementations is required to allow faster modelling of networks, and their resilience strategies;

Performance & scalability: the simulation platform has to be scalable and present good performance to allow faster and larger simulations of realistic network topologies;

Modelling generality: the simulation environment should support the modelling of network components and protocols consisting of mechanisms for resilience that will reside at protocol layers 1–7.

Table I presents a comparison between the different network simulators considering the requirements above. Most simulators offer an extensive library of network models, apart from NS-3, which is still a relatively new endeavour and whose models need to be ported from NS-2 manually, and SSFNet, whose development was discontinued in 2004 and the availability of new protocol models is now limited. Also, NS-2 has been consistently reported to offer limited scalability and performance [20]. Moreover, all simulation environments considered are suitable for modelling general communication networks and protocols at different levels. Based on our evaluation, OPNET, QualNet and OMNeT++ are good candidates for implementing the PReSET toolset, as they score highly in all our evaluation criteria. We chose to implement PReSET using OMNeT++ as we have previous experience with it, and because it is freely available, in contrast to OPNET and QualNet. This latter point makes PReSET available to a wider audience, potentially resulting in it having a greater impact. OMNeT++ is also considered one of the most widely used simulators for research in the area of communication networks [21].

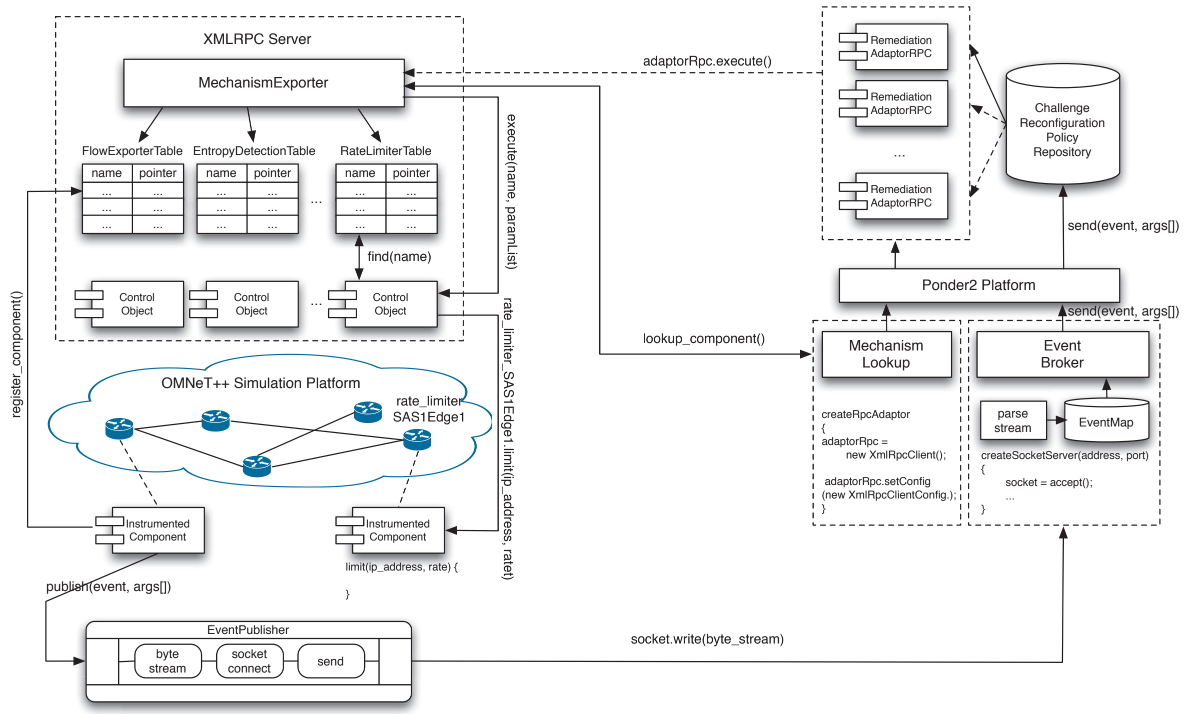
Our work, however, is based on an earlier prototype that we implemented using the SSFNet simulator, which was previously reported in [12]. The lack of new protocol models made SSFNet unsuitable for the evaluation of resilience strategies and the latest network attacks. In this paper, we present a detailed description of the architecture and main components of our OMNeT++-based implementation, and a comprehensive case study and set of experiments that validate the use of the toolset for the evaluation of resilience strategies.

B. Architecture and Main Components

OMNeT++ is a general discrete event simulator that provides the basic machinery and tools to write simulations. In order to support the modelling of communication networks, the INET framework[2] provides extension models for several wired and wireless networking protocols, including UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS and OSPF. OMNeT++ consists of C++ modules that communicate via message passing. Messages are exchanged through input/output gates. Simple modules can be combined in hierarchies in order to build more complex components, called *compound modules* (e.g., mail servers, routers, etc). OMNeT++ also provides tools for designing network topologies (the NED language and editor) and supports plug-in extensions (e.g., a customised event scheduler).

The main motivation for the integration of a policy framework to a simulation environment was to enable the evaluation of the dynamic reconfiguration of network mechanisms in a resilience strategy. Fundamentally, any simulation environment could be used to evaluate hard-wired resilience strategies only, whereas we required the evaluation of strategies that deploy and reconfigure resilience mechanisms on-demand, according to attacks or network conditions that are monitored dynamically. To implement this dynamic behaviour we chose Ponder2 because it can be easily extended with additional functionality. Ponder2 implements a policy execution framework that supports the enforcement of both *obligation* and

[2]http://inet.omnetpp.org



```
limit(ip_address, rate) {

}
```

```
createRpcAdaptor
{
adaptorRpc =
        new XmlRpcClient();

 adaptorRpc.setConfig
(new XmlRpcClientConfig.);
}
```

```
createSocketServer(address, port)
{
        socket = accept();
        ...
}
```

Fig. 2. Policy-driven resilience simulator architecture and main components

authorisation policies. Ponder2 policies are written in terms of user-defined *managed objects*, e.g. adapters for interfacing with real network equipment. In particular, new adapters can be written for third-party components, which in our case are instrumented objects running within the simulation environment. This feature readily allows Ponder2 policies that are evaluated via simulation to be used to control real networked mechanisms with Ponder2 adapter implementations. In the following, we describe the choices available to realise the integration between the simulation environment with the policy-management framework.

1) Integration Techniques: Several techniques to allow the integration between a network simulator environment and external third-party applications were discussed in [22]:

Socket connection: proxies that run in the simulation environment maintain socket connections to external applications. Sockets wait for connections and are responsible for delivering messages from the simulated components to the third-party application, and vice-versa;

Source code integration: this method is straightforward for simple applications, which require that the third-party application needs to be compiled with the simulation. However, this may be difficult for larger applications due to dependencies in the build environment;

Shared libraries: is based on the integration between the simulation tool and the binary code of the third-party application. It is similar to source code integration but avoids problems related to the building process, because the build environments are kept separated.

The integration between OMNeT++ and Ponder2 is based on proxies, which is similar to the *socket connection* method. However, we are using XMLRPC[3] proxy servers running within the simulation instead. Socket-based integration is suitable when the third-party application does not need large volumes of data from lower layer protocols [22]. Instead, in our implementation, exchanges are limited to selected control events and corresponding management commands. This technique may, however, cause CPU scheduling and synchronisation issues since simulations run faster and consume more CPU than applications running in real-time. We expect that these issues can be mitigated because, in contrast to the work presented in [22], we do not exchange packet-level information (large quantity, fast processing) with the policy framework.

2) Implementation: Fig. 2 illustrates the architecture and main components of the policy-driven resilience simulator implementation. Instrumented mechanisms in the simulation environment implement an XMLRPC server through the ***MechanismExporter*** component. This component is used to register and export the management interfaces for the resilience mechanisms available in the simulation. A management interface provides callback functions to management operations that can be used to reconfigure a resilience mechanism, for example, to adjust the throttling rate of a rate limiter. For each type of mechanism, a ***ControlObject*** defines the management functionality to be exported via this management interface, and maps invocations to their respective method implementations on an ***InstrumentedComponent***. This mapping relies on a table $\langle name, pointer \rangle$ that matches different invocations to the correct instance of a specific mechanism.

Whereas the components above implement an XMLRPC

[3]http://xmlrpc-c.sourceforge.net

TABLE II. RESILIENCE MECHANISMS IMPLEMENTED AS OMNET++ MODULES

Module	*Description*
Flow Exporter	This module is used to report flow summary information to a configured sink, such as the *Classifier*. It can be enabled, disabled, and configured with a sampling rate, i.e., the number of packets per time unit used to generate the summaries, and the flow timeout period.
Link Monitor	In OMNeT++ communication links are realised by creating "channels." To implement a module that monitors the utilisation of a link and can trigger an event if a threshold is reached, we extended the `cDatarateChannel` class. This allows us to place a monitoring object in arbitrary locations in a network topology. The Link Monitor is typically used to indicate the onset of a challenge that is causing anomalous traffic volumes.
Rate Limiter	To mitigate challenges that generate an excessive volume of traffic, such as a DDoS attack, we have implemented a module that can shape network traffic. The Rate Limiter module can be configured to probabilistically drop packets from a specified link, those that have a given IP destination address, or (source, destination) IP address and port number tuple.
Entropy Detection	A number of network-borne malicious activity results in a change in entropy of traffic features. For example, this is the case for Internet worm propagation and DoS attacks. Consequently, we have implemented an entropy-based detection module, which monitors the source IP, source port, destination IP, destination port and transport protocol type for changes in entropy. The module computes the entropy of these five features using Shannon's entropy algorithm [23]. A threshold can be defined that triggers an event.
Worm Differentiator	Using signatures based on traffic feature entropy, such as those generated by the Entropy Detection module, this module can identify Internet Worm behaviour that has been previously observed. The module yields the name of an identified Worm or indicates that it has no matching signature.

server for exporting the management functionality to the policy framework, a socket interface has been built to communicate and translate observed events from the simulation environment to the policy framework. Events are used to indicate conditions observed in the simulated network that may require management actions, such as the detection of an attack. The ***EventPublisher*** component is responsible for establishing a connection with a Ponder2 instance and generates events of the form:

$$eventName?arg1 = val1; arg2 = val2; arg3 = val3; ...$$

These events are converted to a byte stream and sent via the socket connection to the Ponder2 instance. At the Ponder2 side, an ***EventBroker*** parses the byte stream received from OMNeT++ and maps it to Ponder2 events. A Ponder2 event may trigger one or more *event-condition-action* (ECA) policies, and the actions specified by a policy define what resilience mechanisms, which are executing within the simulation environment, should be reconfigured and how. References to these mechanisms are obtained via the ***MechanismLookup*** component. When a Ponder2 policy is triggered, actions are invoked using the XMLRPC protocol for the respective mechanism, which is abstracted by an instance of the ***RemediationAdaptorRPC*** component.

C. Types of Attacks and Mechanisms Supported

We have implemented a set of resilience mechanisms as OMNeT++ modules, which can be applied to realise a number of resilience strategies. These modules include the extensions described in Section III-B, thus enabling them to interface with Ponder2. Most of the modules were realised by adapting the standard INET framework *Router* module. A list of the resilience mechanisms that we implemented, alongside a brief description, is presented in Table II.

An example configuration of some of the these modules is shown in Fig. 3 to create an *Enhanced Router* that includes resilience functionality. This can be seen as a form of programmable router, capable of traffic monitoring as well as traffic shaping. In our implementation, the *Entropy Detection* module is positioned above the network layer implementation, and receives packets from it. The *Rate Limiter* resides between the network and physical layers, and thus has access to every incoming and outgoing packet. As mentioned in Table II, the *Worm Differentiator* can be used to identify known worms using entropy measures from the *Entropy Detection* module. Finally, the extended channels that implement the *Link Monitor* can monitor both the traffic traversing the enhanced router via the PPP and Ethernet modules.

To simulate large-scale IP networks and attacks we use the *ReaSE* tool [24], which permits the creation of realistic topologies and the generation of background and attack traffic. Of particular importance for our experiments, it can generate DDoS attack traffic based on the Tribe Flood Network [25]. Furthermore, *ReaSE* can generate Code Red worm [26] propagation behaviour. We have extended the package to simulate the Witty and Slammer/Saphire worms [27], including port scanning behaviour. Resilience strategies for the Code Red and Witty worms were previously discussed in [28]. In the next section, we will present the evaluation of resilience strategies for containing Slammer/Saphire worm propagations.

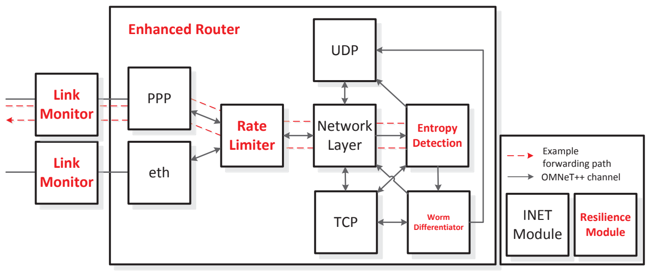


Fig. 3. Example configuration of OMNeT++ modules for network resilience

IV. CASE STUDY AND EXPERIMENTS

Internet worm behaviour is particularly difficult to reproduce for two reasons: *i)* they represent a *large-scale phenomenon*, which requires the model to be of comparable scale for the correct modelling of the propagation dynamics, and *ii)* they occur over *extended time scales*, perhaps with changing infection intensity. Attempts to simplify a model in terms of these factors can lead to incorrect simulations [29]. Consequently, the evaluation of resilience strategies that detect and mitigate Internet worms are an ideal application of our toolset. We now present a worm resilience strategy realised through the co-operation of a number of policy-enabled mechanisms.

A. Worm Simulation

For the case study, we made use of our SQL Slammer worm implementation. As a means of infecting new hosts, this malware continuously sends 404 byte UDP packets to random IP addresses. If a malicious packet infects a new host, it will start sending probing packets. Our simulated network

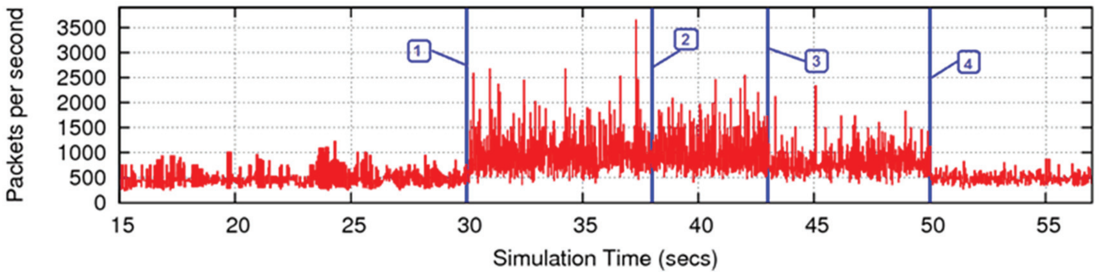


Fig. 4. Simulation results for the worm resilience strategy

consists of 35 Autonomous Systems (ASes): 26 stub ASes connected by 9 transit ASes, 1700 hosts, 80 web servers and 15 interactive servers generate background traffic. A maximum of 10,000 probing packets could be sent from an infected host. Several hosts throughout the network are initially nominated to be zombies, which generate worm probing packets.

Our resilience strategy is simulated at a stub AS and is intended to detect and mitigate the effects of the Slammer worm scanning behaviour. The various resilience mechanisms are activated on a gateway router of the AS and its ingress link from a core router. Initially, a *Link Monitor* module is invoked on each of the ingress links to monitor link utilisation – a threshold parameter is defined, which if exceeded results in an event being generated. An *Entropy Detection* module is also invoked and configured with a list of features that it is to monitor. As mentioned earlier, the *Entropy Detection* module continuously monitors the traffic features' distributions using Shannon's entropy algorithm.

Network anomalies can cause changes in observed IP address or port distributions. If the entropy value of these features increases dramatically, it indicates that the distribution is disperse; if the entropy value decreases sharply, it indicates that the distribution is concentrated around a single IP address or port. The *Entropy Detection* module recomputes entropy for five traffic features every few seconds and stores them in a vector. Anomalies need to be detected as early as possible, so if the recalculation interval is large, we might miss the opportunity to report the anomaly at an early stage. On the other hand, if the interval is too small, resources might be wasted and the computation complexity increased. In our simulation, the *Entropy Detection* module generates an event when worms disturb the entropy values of traffic features beyond a given threshold. Both the entropy recalculation interval and the threshold can be configured by policies.

Another component, the *Worm Differentiator*, maintains a database of *entropy signatures* of known worms. On notification from the *Entropy Detection* module, policies activate the *Worm Differentiator* along with details of the perturbed traffic features. The suspicious traffic features are matched against the signatures of known worms. If a match is found, an event is generated to identify the type of worm. However, if no match is found, a default policy specifies an initial remediation to protect the network. Additionally, the new signature will be added into the worm signature database. On receiving a notification from the *Worm Differentiator* module, another policy configures the *Rate Limiter* to start discarding all packets that conform to the worm's characteristics, thus throttling all propagation packets, without having to identify attacking sources.

Fig. 4 shows the volume of traffic on the ingress link for a simulated, blind scanning Slammer worm attack. At the start, the *Link Monitor* is activated with an alarm threshold set to an increase in average traffic on the link of twice the previous average. The *Entropy Detection* module is also activated to periodically collect packet-level entropy values on five traffic features: destination IP, destination port, source IP, source port and protocol. Fig. 4 shows the worm propagation starting at approximately 30s (1); an alarm is generated by the *Link Monitor* at 37s (2) due to the high volume of traffic on the ingress link. This form of early detection is appropriate for high-volume attacks. A low-volume attack will evade detection by the *Link Monitor*, however, its traffic feature changes could still be captured by the *Entropy Detection* module. The *Entropy Detection* module is interrogated for any significant changes in the five traffic features. The most recent entropy trend for each feature is computed and compared with the previous average entropy. Entropy changes are shown in Fig. 5, which indicates that the destination IP, source IP and source port have become dispersed, and destination port more concentrated. These results are reported back to further analyse the malicious traffic. Based on the traffic feature distribution, the malicious traffic is confirmed as the worm attack at 43s (3).

Policies are used to specify a coarse grain remediation, actioned to initially shape 25% network traffic on the link. The *Worm Differentiator* is then invoked and identifies that the entropy of these traffic feature distributions match a known signature for the Slammer worm. Further analysis is performed to monitor the source IP address for all incoming UDP packets. Source IP addresses appearing at a significantly higher than average frequency are added to a blacklist. Therefore, at 50s (4) another policy is used to reconfigure the *Rate Limiter* to block all probing packets, specified as all UDP packets from blacklisted sources with a destination port 1434, i.e., that used by Slammer to infect hosts.

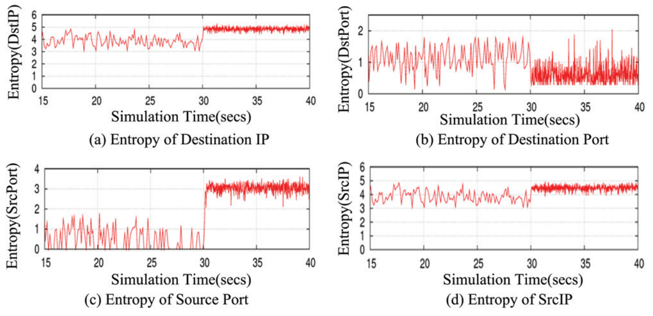


Fig. 5. Entropy changes with the Slammer Worm

B. Evaluation and Discussion

Our platform provides results that show about 21% of benign traffic being blocked during the period of interim mitigation and 28% of malicious worm packets. Off-line risk analysis would determine if the costs (lost benign packets) are appropriate to the benefits (reduced flooding). Our platform provides inputs to such analysis, including the applications that are affected due to the blocking.

A benefit of our approach is the ease with which it enables experimentation and refinement of these trade-offs. We can develop and refine strategies through the configuration of different mechanisms via policies and the online adjustment of parameters such as thresholds. Our platform generates results that enable evaluation of strategies. Further experimentation can identify the trade-offs between early detection and accuracy. New mechanisms can be added to analyse additional features, such as volumes at different levels of granularity or protocol.

V. Related Work

In [21], OMNeT++ was used for evaluating the efficiency of large-scale distributed detection of network attacks, such as DDoS attacks and worm propagation. In particular, OMNeT++ was integrated with *Distack* [30], a framework for the evaluation of detection mechanisms. These mechanisms can be transparently deployed in both real and simulated environments. They are based on a combination of user-written shared libraries that perform basic functions such as packet inspection, filtering, sampling as well as several anomaly detection methods. Likewise, resilience strategies could also be integrated with off-the-shelf detection mechanisms, and an approach similar to the one in [21] could be taken.

In [31], a framework to simulate network attacks and challenges in NS-3 is presented. The framework decouples the specification of a challenge from the network model, allowing different combinations of challenges to be applied to network topologies. Challenges are classified according to their *domain* (wired or wireless), *scope* (nodes, links, area) and *intention* (malicious or non-malicious). To simulate a specific challenge (e.g., a malicious attack or a large-scale disaster), a combination of *links* and *nodes* is disabled in the simulation during a given *interval* – for example, a large-scale disaster is represented by the specification of a certain set of geographic coordinates and subsequently disabling links and nodes within that area. For the purposes of our research, the ability to simulate a range of network challenges is also important, and the framework proposed in [31] might be adapted to our simulation environment.

In [29], SSFNet was used to simulate large-scale Internet worm propagation and its effect on BGP routing traffic. SSFNet was chosen because it enabled scaling up the network model size through parallel and distributed execution, which was essential in modelling large-scale worm propagation. Moreover, SSFNet provides an API for setting up DDoS attack scenarios[4] which could be used to construct attack scenarios to evaluate resilience strategies.

In [32], a modelling tool based on OMNeT++ was presented to permit the analysis of security risks in Supervisory Control and Data Acquisition (SCADA) systems. These systems are used to monitor and control critical infrastructures for electricity, gas, water, waste, railway, and traffic. In particular, the tool for building SCADA simulations supports the integration of simulation components with external devices and applications, thereby allowing the evaluation of the effects of an attack on the physical infrastructure. We are currently assessing the benefits of building a similar integration between simulated resilience mechanisms and physical devices in the network.

In contrast to approaches that use simulations to evaluate the impact that challenges and attack scenarios have on the network [21], [29], [31], [32], we are also interested in evaluating strategies that involve their mitigation. Therefore, our work supports not only the simulation of challenges and algorithms for their *detection*, but also the corresponding activation of mechanisms that will attempt the *remediation* of the effects of a challenge, based on conditions observed during run-time in the simulation. This reflects the main contribution presented in this paper.

VI. Concluding Remarks

This paper presented PReSET[5], a toolset for the evaluation of policy-driven resilience strategies. It is based on an integration between the OMNeT++ simulator and the Ponder2 framework, and can be used to support the evaluation of policy-based management strategies for resilience. Policies are used to orchestrate the behaviour of resilience mechanisms, e.g., anomaly detection and traffic shaping systems, that are realised as modules in the OMNeT++ simulator. Our main contributions are to provide an extensible toolset to model resilience strategies; to allow the offline analysis of a range of anomalies and attack behaviours; and to permit the evaluation of resilience strategies to detect and mitigate security threats. Network simulators have well-known limitations, in the sense that they necessarily abstract some of the details of a real implementation. This may restrict the fidelity for modelling hosts and other components. For this reason, simulations are mostly used during the early stages of development and to model large-scale topologies, for instance. We plan to integrate the toolset with physical network components through a hybrid emulation testbed as part of our future work.

We expect that the toolset presented in this paper will assist network operators in the offline analysis of the impact of network attacks and challenges. Pre-tested configurations can be evaluated and optimal policies may be established before resilience strategies are deployed in the network infrastructure. Furthermore, the same policies used to configure mechanisms in the simulation environment could possibly be used to configure devices in the live network. One of our recent studies [1] shows how resilience strategies that perform optimally in the simulation environment and successfully contain the effects of challenges can be ported to *management patterns*, which are reusable configurations of resilience mechanisms. This allows network researchers and practitioners to capture best practices and effective solutions for network resilience.

[4]http://www.ssfnet.org/javadoc/SSF/App/DDoS/package-summary.html

[5]Available for download: http://www.scc.lancs.ac.uk/PReSET

Acknowledgment

This research is supported by the EPSRC funded India-UK Advanced Technology Centre in Next Generation Networking, by the European Union Research Framework Programme 7 via the PRECYSE project with contract number FP7-SEC-2012-1-285181, and by NICTA (National ICT Australia).

References

[1] A. Schaeffer-Filho, P. Smith, A. Mauthe, D. Hutchison, Y. Yu, and M. Fry, "A framework for the design and evaluation of network resilience management," in *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*. Maui, Hawaii, USA: IEEE Computer Society, April 2012, pp. 401–408.

[2] K. Twidle, E. Lupu, N. Dulay, and M. Sloman, "Ponder2 - a policy environment for autonomous pervasive systems," in *POLICY '08: IEEE Workshop on Policies for Distributed Systems and Networks*. Palisades, NY, USA: IEEE Computer Society, 2008, pp. 245–246.

[3] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *SIMUTools '08: Proceedings of the 1st International Conference on Simulation Tools and Techniques*. Marseille, France: ICST, 2008, pp. 1–10.

[4] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, vol. 54, no. 8, pp. 1245–1265, June 2010.

[5] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Comput. Surv.*, vol. 39, no. 1, p. 3, 2007.

[6] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *IEEE Communications Surveys Tutorials*, vol. 10, no. 1, pp. 20–35, 2008.

[7] PandLabs, "PandaLabs Annual Report 2011 Summary," Panda Security, Tech. Rep., 2011. [Online]. Available: http://press.pandasecurity.com/wp-content/uploads/2012/01/Annual-Report-PandaLabs-2011.pdf

[8] Y. Yu, M. Fry, A. Schaeffer-Filho, P. Smith, and D. Hutchison, "An adaptive approach to network resilience: Evolving challenge detection and mitigation," in *DRCN'11: 8th International Workshop on Design of Reliable Communication Networks*, Krakow, Poland, October 2011, pp. 172 –179.

[9] T. A. Limoncelli, "OpenFlow: a radical new idea in networking," *Commun. ACM*, vol. 55, no. 8, pp. 42–47, Aug. 2012.

[10] P. Smith, A. Schaeffer-Filho, A. Ali, M. Scholler, N. Kheir, A. Mauthe, and D. Hutchison, "Strategies for network resilience: Capitalising on policies," in *4th International Conference on Autonomous Infrastructure, Management and Security (AIMS)*. Zurich, Switzerland: LNCS, June 2010, pp. 118–122.

[11] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "ns-3 project goals," in *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*. New York, NY, USA: ACM, 2006, p. 13.

[12] A. Schaeffer-Filho, P. Smith, and A. Mauthe, "Policy-driven network simulation: a resilience case study," in *SAC'11: 26th Symposium on Applied Computing*. Taichung, Taiwan: ACM, March 2011, pp. 492–497.

[13] M. Sloman and E. Lupu, "Security and management policy specification," *IEEE Network*, vol. 16, no. 2, pp. 10–19, Mar.-Apr. 2002.

[14] NS-2 Website, "The Network Simulator - NS-2," http://www.isi.edu/nsnam/ns/. Accessed in August 2012.

[15] NS-3 Website, "The NS-3 network simulator," http://www.nsnam.org/. Accessed in August 2012.

[16] OMNeT++ Website, "OMNeT++," http://www.omnetpp.org/. Accessed in August 2012.

[17] SSFNet Website, "Modeling the Global Internet," http://www.ssfnet.org/. Accessed in August 2012.

[18] OPNET Website, "OPNET Modeler Accelerating Network R&D (Network Simulation)," http://www.opnet.com/solutions/network_rd/modeler.html. Accessed in August 2012.

[19] QualNet Website, "QualNet," http://www.scalable-networks.com/content/products/qualnet. Accessed in January 2013.

[20] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators," in *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*. New York, NY, USA: ACM, 2002, pp. 38–43.

[21] T. Gamer and C. P. Mayer, "Large-scale evaluation of distributed attack detection," in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–8.

[22] C. P. Mayer and T. Gamer, "Integrating real world applications into OMNeT++," Institute of Telematics, Universität Karlsruhe (TH), Telematics Technical Report TM-2008-2, Feb. 2008. [Online]. Available: http://doc.tm.uka.de/2008/TM-2008-2.pdf

[23] Z. Chen and C. Ji, "An information-theoretic view of network-aware malware attacks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 530–541, September 2009.

[24] T. Gamer and M. Scharf, "Realistic simulation environments for ip-based networks," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–7.

[25] D. Dittrich, "The tribe flood network distributed denial of service attack tool," University of Washington, Tech. Rep., October 1999.

[26] C. C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 138–147. [Online]. Available: http://doi.acm.org/10.1145/586110.586130

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

[28] Y. Yu, M. Fry, B. Plattner, P. Smith, and A. Schaeffer-Filho, "Resilience strategies for networked malware detection and remediation (to appear)," in *6th International Conference on Network and System Security (NSS 2012)*. Wu Yi Shan, Fujian, China: Springer, November 2012.

[29] M. Liljenstam, Y. Yuan, B. J. Premore, and D. Nicol, "A Mixed Abstraction Level Simulation Model of Large-Scale Internet Worm Infestations," in *MASCOTS '02: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*. Washington, DC, USA: IEEE Computer Society, 2002, p. 109.

[30] T. Gamer, C. P. Mayer, and M. Zitterbart, "Distack – a framework for anomaly-based large-scale attack detection," in *SECURWARE '08: Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 34–40.

[31] E. K. Çetinkaya, D. Broyles, A. Dandekar, S. Srinivasan, and J. P. Sterbenz, "A Comprehensive Framework to Simulate Network Attacks and Challenges," in *RNDM'10 - Second International Workshop on Reliable Networks Design and Modeling*, Moscow, Russia, October 2010, pp. 538–544.

[32] C. Queiroz, A. Mahmood, and Z. Tari, "SCADASim - A Framework for Building SCADA Simulations," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 589 –597, dec. 2011.