Analysis of Software Defined Networking defences against Distributed Denial of Service attacks

Ralph Koning, Ben de Graaff, Cees de Laat, Robert Meijer, Paola Grosso
System and Network Engineering group (SNE) University of Amsterdam, The Netherlands
Email: r.koning@uva.nl, b.degraaff@uva.nl, delaat@uva.nl, r.j.meijer@uva.nl, p.grosso@uva.nl

***Abstract*—The Secure Autonomous Response Networks (SARNET) framework introduces a mechanism to respond autonomously to security attacks in Software Defined Networks (SDN). Still the range of responses possible and their effectiveness need to be properly evaluated such that the decision making process and the self-learning capability of such systems are optimized. To this purpose we developed a touch-table driven interactive SARNET prototype, named VNET, and we demonstrated its use through real-time monitoring and control of real and virtualised networks. By observing users interacting with the system at SC15 in Austin, we concluded that in a SDN it is possible to achieve high effectiveness of responses by carefully choosing a relatively minor number of actions.**

I. SARNET FRAMEWORK

Software Defined Networks (SDN) have been proposed as an effective way to build and support secure (network) services. The underlying assumption is that the capability of programming the topology and the paths taken by traffic flows will build stronger and more resilient networks, and provide automated responses in the case of attacks. Therefore, we decided to build a framework that addresses two challenges that are currently not covered by other SDN systems. Firstly, we want to provide a system that can react **autonomously** to attacks by exploiting a knowledge base of tactics tailored to the strategies defined by the businesses that use the system. Secondly, we want to allow services to span **multiple domains** by allowing the definition of joint strategies amongst cooperating organisations.

The SARNET framework [1] will provide autonomous response across multiple domains to network attacks by exploiting the underlying SDNs functionalities and virtualised network functions. The vision is that the SARNET framework will be adopted by enterprises to provide secure (cloud) services. The autonomous response in a SARNET is achieved by means of a control loop, depicted in Fig. 1.

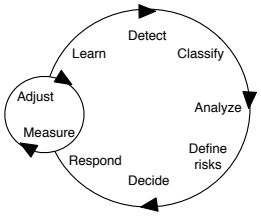

Fig. 1. The SARNET control loop

Every SARNET will monitor the state of the network and services by continuously evaluating a number of *security observables*. Detection of violation of the expected state and values of these observables will initiate the control loop. After a recognition phase (*classify, analyse,* and *risk*), a SARNET will autonomously *decide* the appropriate response to bring the network back to an acceptable security state. Adjustments might be needed if the observables do not return to the desired state after responding. A SARNET will reprogram the network flows, redefine the location of the virtualised network functions, and possibly move the location of computing and storage services.

While developing the framework, it is imperative to assess, via fast prototyping, what the implementation issues are, and possibly whether architectural assumptions need revision. Therefore, we set out to investigate:

- What is the most appropriate way to expose the security observable to external components, either human or software? Concretely, which visualization techniques are suitable for SARNETs?
- What is the range of responses possible in a SARNET and how do these depend on the underlying SDN control software?
- What are the metrics that can guide the selection of responses to attacks during the *decide* phase, and what are the most valuable metrics we can store in the *learn* phase to determine solutions' effectiveness for future selection?

In this paper, we present the results and findings on the above questions that we obtained with our interactive prototype, where visitors use a multi-touch interface to detect and respond to DDoS attacks, and was demonstrated during the Super Computing conference held in Austin, TX in November 2015 (SC15).

II. PROTOTYPE ARCHITECTURE

We developed a prototype called VNET that supports an initial number of SARNET control loop elements, with particular focus on *Detect, Decide, Respond, Measure,* and *Learn* phases. Currently, VNET is able to provide a visualisation of a network suffering from basic DDoS attacks and it allows users to manipulate the network characteristics with direct visual feedback on their actions and the effects thereof. It allows the creation of simple observables based on the current state of the network topology, traffic and elements. Additionally, VNET allows scripting of attack scenarios, which execute network changes using the network controller. Real-time monitoring data and observable states are forwarded to

the UI for visualization and user interaction. Fig. 2 shows the application components of the VNET.

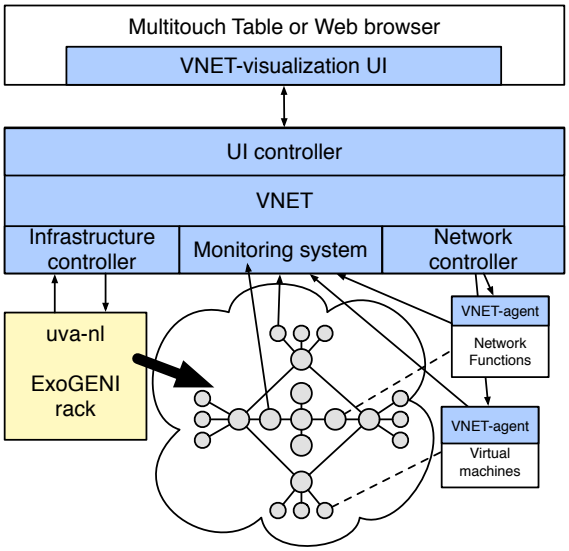

Fig. 2. Software components in the VNET prototype.

A. ExoGENI

As initial underlying platform for the VNET operations, we used ExoGENI [2]. ExoGENI is a platform for orchestrating cloud resources (OpenStack), SDNs (OpenFlow) [3], and network circuits [4]. There are currently about 20 ExoGENI racks operational at various educational and research institutes, including one located at the University of Amsterdam (the *uva-nl rack*). Resources are deployed in the form of slices and can span multiple racks and networks (physical sites). The platform hides the hardware differences between the racks and automatically configures the network. Network elements and functions are implemented as virtual machines.

B. Network functions

Independence of the underlying infrastructure was an important design requirement. Therefore, we use Ansible[1] playbooks to build our network functions and to prepare the necessary VM images. These playbooks contain instructions to install each virtual machine and to configure the required software packages, including the VNET agent which we use to monitor the VM. We currently have network functions for traditional and OpenFlow switches, RIP and OSPF routers, and SDN controllers such as OpenDaylight and Ryu [5]. These network functions are used in the network topologies that are deployed by VNET.

C. Infrastructure controller

The infrastructure controller acquires and monitors the topology from the underlying infrastructure. This topology consists of the VMs and virtual network links. It translates the topology data from the Infrastructure as a Service (IaaS) controller and converts this to the VNET internal format. The topology is regularly polled at a tunable rate dependent on the expected frequency of changes. A push based approach can also be used if the IaaS controller supports custom plugins or sending topology updates.

D. Monitoring controller

The purpose of the monitoring controller is to collect information, metrics, and statistics from the nodes and links in the network and to pass this to the VNET interface. The node and network state (bandwidth usage and link state per interface) and various function specific data (e.g. spanning-tree information) is sent over an encrypted channel by the VNET agent that runs on the VMs.

IaaS platforms do not always accurately provide topology information. In those cases, the monitoring controller is capable of deducing the topology based on platform specific meta-data provided by the VNET agent. For example, in case of ExoGENI, this meta-data contains the URN that uniquely identifies the node, the name of the slice, and the cluster worker node the VM is running on. ExoGENI however, does not provide a URN representation for its interfaces to the VMs; therefore, the monitoring controller uses the interface IP addresses provided by the VNET agent to construct the final topology.

E. Network controller

VNET interacts with the network components and changes properties that alter the behaviour of the network and the traffic flows. The network controller facilitates this; it uses an RPC channel to the VNET agents to set node and link configuration, and to execute commands on the nodes. Changes to the network can be triggered by the user via the multi touch interface or automatically by VNET.

F. VNET agent

The VNET agent monitors and controls the node. The daemon maintains secure connections to both the monitoring controller, and the network controller over a TLS secured WebSocket[2] connection. The agent reports detailed interface statistics and host meta-data such as host name, interfaces and IP addresses. Network function specific data can supplement the host meta-data by writing this data in the form of (JSON) key/value pairs to a predefined directory which will be automatically picked up and transmitted to the monitoring controller.

G. Visualization user interface

The VNET user interface supports both touch and pointer events, it is build in JavaScript using the D3.js[3] library and uses WebSockets to talk to the user interface controller component.

[1] Ansible: https://www.ansible.com/

[2] The WebSocket protocol: https://tools.ietf.org/html/rfc6455

[3] D3.js Data-Driven Documents: https://d3js.org/

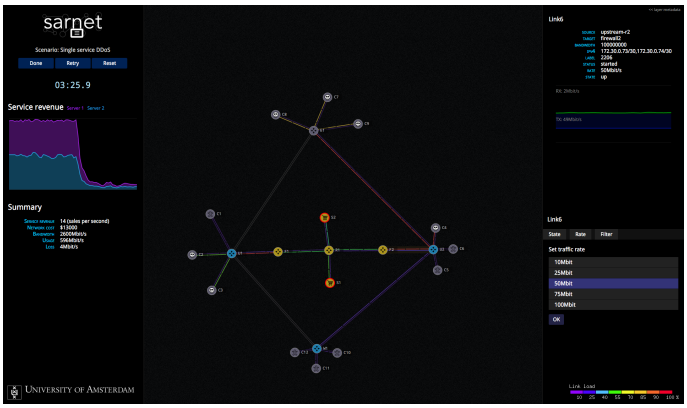

Fig. 3. VNET demonstration running scenario: 2

Fig. 3 shows the VNET user interface, as during the demonstration at SC15. Information is organised in three columns. The left column shows information relevant to the operation of the network; the demo showed scenario controls and service revenue as described in Sec. III-B. The centre column shows a network representation. It displays disabled links and the line colour changes based on link utilization for active links. An observable, node health status, is displayed using a red, orange, or green circle. Finally, the right column shows details of the currently selected node or link, and provides controls for node actions.

The concurrent display of network and service information is an essential element for the SARNET operation, as this is the only way in which the system can maintain the proper view and balance between effect of software defined network operation and the resulting service to the end users.

H. Bootstrapping

To bootstrap a network, we use a topology description, a JSON file with a list of nodes, node type and a list of all the available network links between these nodes. The format is kept basic for readability and ease of use. This is done by using common defaults so properties do not have to be defined for each element. When supported by the underlying infrastructure, multiple domains can be used by simply specifying the location (domain) of the node in the topology. This simplified topology is converted into the appropriate orchestration request for the underlying virtualisation platform. In the case of ExoGENI this is NDL-OWL [6], [7], [8].

I. Scenarios

After bootstrapping the network, we load an attack scenario. The definition consists of two parts. First, the initial state of the network is defined, such as which links are enabled, what their bandwidth is, and which filters are applied. The scenario also defines the visual elements such as the colours or icons of nodes in the topology. Secondly, it allows scripting predefined attack patterns over a period of time. The script contains a list of time points that define when and by which nodes the DDoS attacks are started. Commands contain the target of the attack, its type (e.g. UDP or TCP), the duration of the attack, and its strength in terms of bandwidth.

III. Multi-touch table demonstration

The demo we showed at the Super Computing 2015 (SC15) conference relies on the prototype described in Sec. II. The visitor of the demo was presented with a multi-touch table interface showing an interactive visualisation of a network. The goal was to use this interface to reconfigure the network and minimise the effect of the DDoS attack congesting the service and to recover revenue. By detecting, analysing, deciding and reacting to the attack, the visitor effectively acts as the SARNET control loop (Fig. 1). The 25 VMs used for this demo were hosted on the *uva-nl* ExoGENI rack and the links between the nodes were requested with a maximum bandwidth of 100 mbit/s.

During SC15, the ExoGENI IaaS platform posed some limitations: first, running slices could not be modified[4]. We implemented our own mechanisms to scale bandwidth on the virtual links using *tc*[5] on the interfaces of the virtual machines. We used token bucket filter to shape the outgoing rate to the bandwidth requested by the visitor. Additionally, removing links in an active slice, was implemented by shutting down interfaces on the virtual machines. Secondly, there was no mapping between the interfaces on the virtual machines and the interfaces/links in the ExoGENI topology. We solved this by using a static IP addressing scheme that allowed us to unambiguously identify links and interfaces.

We used three types of network elements to build the demo, *routers* running OSPF, *services* and *customers* that can turn into an *attacker*. *Services* run a web service that simulates web shop transactions; *customers* send transactions to the web service while *attackers* send both transactions and attack traffic to the web services.

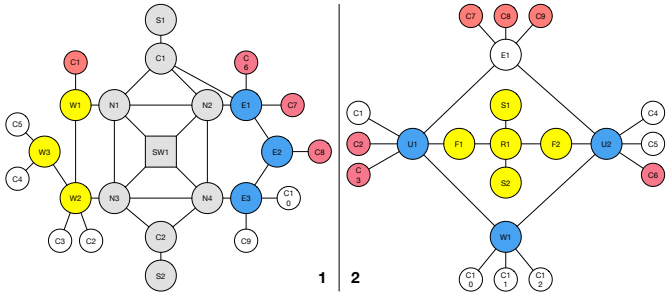

Fig. 4. The two demo scenarios (scenario 1 left and scenario 2 right); colours represent different domains

Fig. 4 shows the two demo scenarios we used during SC15 where we varied the number of elements present, the network topology among them, and the number of domains present. The network topology is pre-programmed to ensure a correct mix and spread of attack traffic such that the problem is solvable by the visitor and ensure that all the defence strategies have effect.

In both scenarios in Fig. 4 virtual customers, C1-12, attempt to perform transactions with two web services S1-S2. The transactions traverse a network consisting of the routers

[4]RENCI deployed slice modification for ExoGENI and deployed this early 2016.

[5]Tc is a tool to configure Traffic Control in the Linux kernel.

W1-3, N1-4, E1-3, U1-2 and F1-2. Scenario 1 also includes a switch in the middle, SW1. Some of these customers are assigned the additional role of attacker and try to congest the network such that the virtual customers will be unable to make transactions to the web services. The dual role of both attacker and consumer is realistic since attack traffic almost always originates from networks that also send legitimate traffic.

Revenue is determined by summing the successful transactions between customer and web service in the network. When the attacks start, the visitor sees that the revenue graph decreases and changes in link utilization. Congestion due to attacks causes links to change colour and eventually, since traffic cannot reach the web services, the web service icon becomes red as well. This is implemented by using an observable on the amount of sales handled by the service. When the sales drop below a certain threshold, the observable triggers and changes the web service symbol in the user interface changes from green to red. When the service recovers, the visualization turns back to green. The attack traffic consists of UDP *iperf2*[6] traffic from multiple hosts at different rates.

A. Responses and costs

We implemented four responses that can be applied on all links between network elements: 1) link *state*, to shut down links; 2) *rate up*, to scale bandwidth upward; 3) *rate down*, to scale bandwidth downward; 4) *filter*, to filter out attack traffic. Based on the network display, the visitor can apply one or more of these methods on congested links to restore revenue. The operations have associated costs which are determined as follows:

$$cost = b\frac{\sum_i r_i}{2} + f\sum_i a_i$$

- where b is interface cost in $ per megabit; we used $b = 10$
- where i is an active interface
- where r_i is link bandwidth of interface i in Mbit/s
- where f is the cost of a filter in $; we used $f = 500$
- where a_i is 1 if a filter is active on interface i or 0

The value for b was chosen based on the consideration that bandwidth cost in North-America is $10 per megabit per second [9]. The value for the parameter f derives from the observation that the market offers DDoS mitigation services from a few hundred to thousands of dollars per month; we therefore chose $500 for a filter action since it is 100% effective when applied to a link. Because in this demonstration the number of nodes is fixed, we did not add node costs to the equation.

B. Collecting solutions

To collect defence strategies from all visitors operating the system, we limited the time for each visitor to 4 minutes; during this time, the visitor's goal is to minimise the attack and maximise the sales. Fig. 3 on the top-left, shows the controls for the demonstration. Reset stops the attack and starts with a clean scenario. *Start/Retry* lets the visitor retry the problem without submitting. *Done/Submit* ends the timer and submits the final result. When the visitor presses *done*, the revenue over a small time window is measured and this is considered the final score. *Submit* will save the solution.

Per visitor we stored a unique session id and the information listed in Table I and II.

[6]We used iperf2 instead of the newer iperf3 because iperf2 does not require a control channel for UDP. https://sourceforge.net/projects/iperf/

IV. Results

The submissions of the users can be used as a dataset to possibly improve automated response to the problems presented in the demonstration. The solutions provided by the visitors in combination with our observations during the demonstration give insight in which attack responses might be effective.

Table I and II show the solutions collected during SC15 for the two scenarios described in section III; *rank* shows the best solution, *rank* is currently based on the revenue *recovery* − network *cost* where *recovery* is the percentage of the revenue that is recovered by the given solution and *cost* is the percentage of the original network cost. *changes* shows the changes to the topology which is the sum of the actions: link *state* (link down), *rate up* (increasing bandwidth), *rate down* (decreasing bandwidth), and traffic *filter*s.

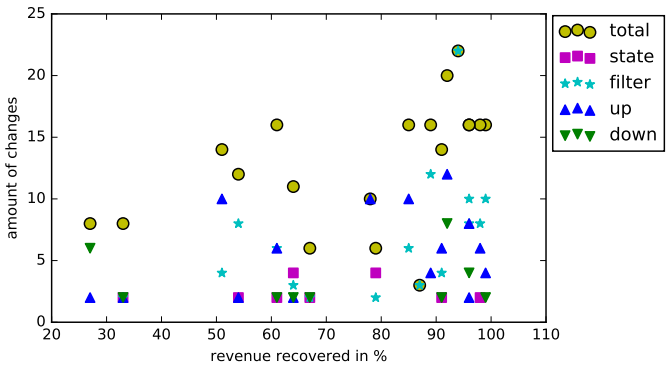

Fig. 5. Relation between the amount of topology changes and revenue recovery in scenario 1

TABLE I. Collected data during demonstration: scenario 1

rank	recovery	cost	changes	state	rate up	rate down	filter
1	92	103	20	0	12	8	0
2	91	112	14	2	6	2	4
3	87	108	3	0	0	0	3
4	79	100	6	4	0	0	2
5	98	126	16	2	6	0	8
6	96	127	16	0	2	4	10
7	99	132	16	0	4	2	10
8	67	100	6	2	0	2	2
9	78	114	10	0	10	0	0
10	96	133	16	0	8	0	8
11	64	103	11	4	2	2	3
12	85	129	16	0	10	0	6
13	89	138	16	0	4	0	12
14	61	119	16	2	6	2	6
15	27	94	8	0	2	6	0
16	54	123	12	2	2	0	8
17	94	164	22	0	0	0	22
18	33	103	8	2	2	2	2
19	51	126	14	0	10	0	4

Fig. 5 and Table I show that filters are used in most of the submissions (16 of 19). The best ranked solution uses no filters and only link rate changes, which are less costly. However, the application of filters seem to have a significant effect on revenue. Analysis confirmed the significance of $p = .049$ and the adjusted R^2 of .162 indicates that filters have a positive effect on the revenue.

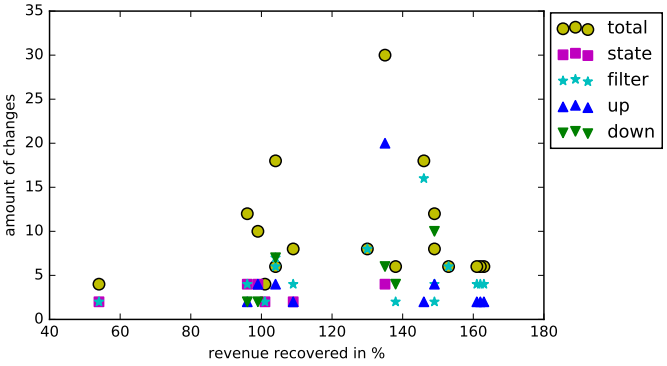

Fig. 6. Relation between the amount of topology changes and revenue recovery in scenario 2

TABLE II. COLLECTED DATA DURING DEMONSTRATION: SCENARIO 2

rank	recovery	cost	changes	state	rate up	rate down	filter
1	149	98	12	0	0	10	2
2	163	119	6	0	2	0	4
3	162	119	6	0	2	0	4
4	161	119	6	0	2	0	4
5	138	97	6	0	0	4	2
6	153	123	6	0	0	0	6
7	149	121	8	0	4	0	4
8	135	121	30	4	20	6	0
9	99	96	10	4	4	2	0
10	130	130	8	0	0	0	8
11	101	103	4	2	0	0	2
12	109	115	8	2	2	0	4
13	96	106	12	4	2	2	4
14	146	165	18	0	2	0	16
15	104	123	6	0	0	0	6
16	104	123	18	0	4	7	7
17	54	103	4	2	0	0	2

In Fig. 6 and Table II, most of the solutions result in a recovery above 100%. This is possible because of the initial state of the topology where the two links to the web services were congested by consumer requests. By increasing the bandwidth on these two links, more transactions can reach the web services resulting in revenue gain. In scenario 2, 15 of the 17 solutions use filters; in contrast to scenario 1, filters did not show a significant effect on the revenue. Analysis of the three similar solutions (rank 2, 3, 4) in II showed that they were submitted by different users at different times and the changes were applied to different nodes in the network.

Fig. 7 suggests a positive correlation between revenue gain due to the solution and solution costs. Further data analysis showed that the correlation is moderate in scenario 1, $r = .511$, which is significant on a 2-tailed .025 level. For scenario 2, the correlation is weak, $r = .356$, and not significant with a p-value of .161. The figure also shows that most solutions for network 2 provide revenue gain and that it is possible to achieve both revenue gain and network cost reduction simultaneously.

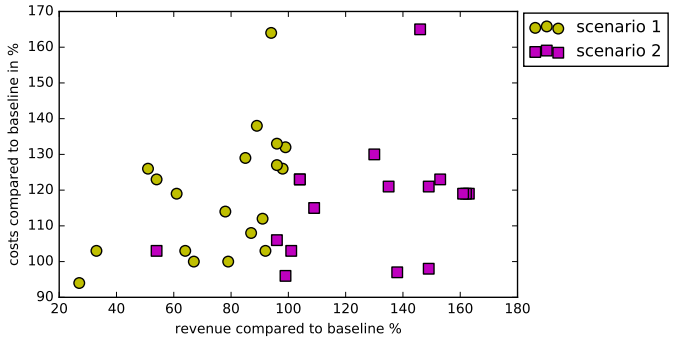

Fig. 7. Relation between cost increase and revenue increase of a solution

V. DISCUSSION

If we look back at the four types of responses that we can perform (see Sec.III-A), we can draw some general conclusions.

Scaling bandwidth upward or downward requires a careful analysis of the distribution of legitimate network traffic and malicious traffic. In general, upward scaling should be deployed when the attack traffic cannot consume the additional bandwidth, which is then available for legitimate traffic only. This strategy works only when the subsequent downstream links are of equal or higher capacity. On the other hand, scaling downward is the strategy of choice when a large amount of attack traffic traverses a certain link. This needs to happen as close as possible to the malicious traffic sources.

Filtering actions were very effective especially in scenario 1, but also admittedly simple to perform as our attack traffic was always UDP and our legitimate traffic TCP. In case of more sophisticated attacks, defining filter patterns will be more complicated; and the execution of complex patterns may require specialized anti DDoS solutions which can be costly especially if one has to filter the higher layers. On the other hand, shutting down links proved to be much less effective. Routing algorithms were, in this case, counter-productive, as they would re-route the attack traffic via other paths in the network, once a link was disabled. These two extremes highlight the need for a proper balance between the traditional routing goal of general availability and the fine-grained behaviour required to fend off attacks.

In scenario 2, the revenue gain is more dependent on the skill set of the visitor since from the results there was no correlation between specific changes and revenue gain. This also showed in practise; combinations of responses were effective but required the user to have advanced experience with network engineering. By tactically shutting down links, we were able to reroute most malicious traffic via one path and legitimate traffic over the other; subsequently, by applying bandwidth scaling, up in case of legitimate and down in case of malicious, we were able to restore most of the revenue without applying expensive filtering. Applying these methods results in temporary loss of traffic, but we did not consider this a major drawback as we were not looking for an approach that minimizes impact on the traffic since when under an attack impairment is already the case.

We learned that SDN platforms are suitable for security

purposes only if they provide an extensive set of primitives available to programmers. In our case, we used the ExoGENI platform as underlying network infrastructure. We discovered that the network slices we could create would not exhibit the required flexibility to effectively defend against network attacks. For example, it was not possible to add or remove links or nodes in an existing ExoGENI topology. This implies that the instruction set exposed by the network controllers will limit the response capabilities of the SARNET framework.

Finally, as network topologies increase in size, the choice of the optimal strategy and the decision on where to act will become too complex for a manual solution. Autonomous frameworks such as SARNET will therefore provide the necessary aid to automate the response.

The results showed a positive correlation between the revenue gain and the use of filters for scenario 1 but not in scenario 2. For all the other changes, we found no correlation; therefore, without further research, we cannot say anything about the effectiveness of the countermeasures because they may be dependent on the specific scenario. Effectiveness of the solution is in this demonstration determined by subtracting the solution cost from the revenue gain which is sufficient for demo purposes. Yet, for real situations other factors may need to be included, e.g. resiliency to network failure.

VI. Related work

While security can be enhanced by using SDNs, it is also true that such networks bring in specific attack possibilities. In [10], Scott et al. clearly explain that the features of SDN that are appealing and useful to enhance security are at the same time the ones that can expose these type of networks to novel types of attacks. Sezer et al. [11] compiled an extensive overview of the implementation challenges for adoption of SDN and included a security overview of the possible vulnerabilities. These works teach us that a framework like SARNET will need to include the knowledge of the peculiarity of SDN's attack surfaces when compiling the network topology to be instantiated and limit its exposure to these new attacks. Likewise SDN-specific attacks need to be part of the SARNET knowledge base used in the control loop during the classification phase.

Visualization of network behaviour exists in many prototypes. Network weather-maps like [12] are a powerful visualisation method. Still, combinations of monitoring and controlling SDN networks are not yet mainstream and VNET shows how to accomplish both in a simple and intuitive manner. Furthermore, the simultaneous exposure of service and network information is a first step toward a tighter integration of network and applications.

VII. Conclusion and future work

The SARNET prototype, VNET, and the demonstration at SC15 led to insights in what factors are necessary to autonomously defend against cyber-attacks. We showed that concurrent display of network and service information provide the visual aid required for human analysis of DDoS attacks. Automatic response requires a measure of effectiveness, calculating this by subtracting cost from revenue proved to be sufficient for this demonstration. Additionally, the results showed that effective solutions can be achieved using a small number of countermeasures.

More research is required to determine which additional factors are necessary to calculate effectiveness and cost of a countermeasure. Response actions are limited by the API of the underlying SDN platform, which, in this prototype, prevented us from adding and removing network elements. As IaaS platforms differ in programmability, the range of response actions also differ per provider. Including this factor in calculating effectiveness will be an important next step, especially when progressing to multi-domain response strategies.

Acknowledgements

SARNET is funded by the Dutch Science Foundation NWO (grant no: CYBSEC.14.003 / 618.001.016) and the National project COMMIT (WP20.11) Special thanks to CIENA for hosting our demonstration at their booth at SC15 and our other research partners TNO and KLM. Additionally, we thank the ExoGENI team at RENCI for their support and prompt response in resolving issues involving their platform.

References

[1] SNE group – University of Amsterdam. (2016, Feb.) SARNET. [Online]. Available: https://sarnet.uvalight.net/

[2] I. Baldine, Y. Xin, A. Mandal, P. Ruth, C. Heerman, and J. Chase, "Exogeni: A multi-domain infrastructure-as-a-service testbed," in *Testbeds and Research Infrastructure. Development of Networks and Communities*. Springer, 2012, pp. 97–113.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] I. Baldine, Y. Xin, A. Mandal, C. H. Renci, J. Chase, V. Marupadi, A. Yumerefendi, and D. Irwin, "Networked cloud orchestration: a geni perspective," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 573–578.

[5] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (sdn) controllers," in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. IEEE, 2014, pp. 1–7.

[6] J. Van Der Ham, F. Dijkstra, P. Grosso, R. Van Der Pol, A. Toonk, and C. De Laat, "A distributed topology information system for optical networks based on the semantic web," *Optical Switching and Networking*, vol. 5, no. 2, pp. 85–93, 2008.

[7] R. Koning, P. Grosso, and C. de Laat, "Using ontologies for resource description in the cinegrid exchange," *Future Generation Computer Systems*, vol. 27, no. 7, pp. 960–965, 2011.

[8] M. Ghijsen, J. Van der Ham, P. Grosso, and C. De Laat, "Towards an infrastructure description language for modeling computing infrastructures," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*. IEEE, 2012, pp. 207–214.

[9] Matthew Prince – CloudFlare. (2016, Feb.) The relative cost of bandwidth around the world. [Online]. Available: https://blog.cloudflare.com/the-relative-cost-of-bandwidth-around-the-world/

[10] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN Security: A Survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, Nov 2013, pp. 1–7.

[11] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 36–43, July 2013.

[12] Network Weathermap. (2016, Feb.). [Online]. Available: http://network-weathermap.com/