SDN4S: Software Defined Networking for Security

Koulouris, Theofrastos; Casassa Mont, Marco; Arnell, Simon

Hewlett Packard Labs

HPE-2017-07

Keyword(s):
SDN; Big Data for Security; Threat Detection; Threat Remediation; Security

Abstract:
Security Operations Centers (SOCs) rely on analysts to perform largely manual processes to carry out the various stages of the incident management lifecycle. These processes are time-intensive and typically require much context switching and hand-off between monitoring and operations analysts, introducing considerable delays into the resolution of incidents. With enterprise networks facing malware threats of increasing complexity and volume, this approach becomes unsustainable. It is crucial, therefore, to develop solutions that dependably automate and accelerate incident management tasks and only involve the limited pool of highly-trained and experienced analysts an organization can have at its disposal when truly necessary, where it matters. In this report we introduce SDN4S: a system and solution to minimize the time between incident detection and resolution by using automated countermeasures based on Software-Defined Networking (SDN). SDN4S creates incident-specific response workflows orchestrating actions and network-based countermeasures automatically upon receiving an alert, leading to faster and more predictable incident response. We describe the architecture and implementation of SDN4S, and report on the test deployment of the system on our research network.

External Posting Date: January 23, 2017 [Fulltext]
Internal Posting Date: January 23, 2017 [Fulltext]

SDN4S: Software Defined Networking for Security

Theo Koulouris, Marco Casassa-Mont
Hewlett Packard Labs
Hewlett Packard Enterprise
Bristol, UK
theo.koulouris@hpe.com, marco_casassa-mont@hpe.com

Simon Arnell
Enterprise Security Services
Hewlett Packard Enterprise
Bristol, UK
simon.arnell@hpe.com

***Abstract*— Security Operations Centers (SOCs) rely on analysts to perform largely manual processes to carry out the various stages of the incident management lifecycle. These processes are time-intensive and typically require much context switching and hand-off between monitoring and operations analysts, introducing considerable delays into the resolution of incidents. With enterprise networks facing malware threats of increasing complexity and volume, this approach becomes unsustainable. It is crucial, therefore, to develop solutions that dependably automate and accelerate incident management tasks and only involve the limited pool of highly-trained and experienced analysts an organization can have at its disposal when truly necessary, where it matters.**

In this report we introduce SDN4S: a system and solution to minimize the time between incident detection and resolution by using automated countermeasures based on Software-Defined Networking (SDN). SDN4S creates incident-specific response workflows orchestrating actions and network-based countermeasures automatically upon receiving an alert, leading to faster and more predictable incident response. We describe the architecture and implementation of SDN4S, and report on the test deployment of the system on our research network.

Keywords—incident response; SDN; security; remediation; countermeasures; playbook

I. INTRODUCTION AND MOTIVATION

IT infrastructure has become one of the most valuable assets of any organization, crucial to its everyday operations. Recognizing the potential cost a security incident may incur to their business due to disruption of operations, loss of intellectual property, leakage of confidential information, legal liability or damage to reputation, a growing number of organizations set up computer security incident response teams (CSIRTs, also known as Computer Emergency Response Teams - CERTs) tasked with handling security incidents and managing the overall security of their networks.

Incident response teams face numerous challenges in carrying out their duties. The typical process for handling security incidents is largely manual, labor-intensive and requires considerable context switching and hand-off between various teams and individual analysts (Figure 1), introducing significant delays into the overall incident resolution process (e.g. [1,2]). Even though analysts use various tools and scripts to assist them at the various stages of the incident response process, the resulting workflows are largely ad-hoc, dependent on the types of vendor-specific functions and features supported by each piece of equipment investigated and susceptible to blocking while sub-tasks assigned to other teams are in progress or higher-priority cases take precedence.

To complicate matters, enterprise networks face ever increasing numbers of threats of evolving complexity. Organizations are responding by deploying an increasing array of SIEM (Security Information and Event Management) and IDS (Intrusion Detection System) systems which in turn generate even more alerts for security teams to investigate, further compounding the problem. Although such systems can be configured to filter out false positives caused by legitimate but unusual behavior or misconfigurations, the increased diversity of today's traffic makes it difficult to understand (and teach machines) what constitutes "normal" traffic and what is unusual. As a result of this "semantic gap" [3], alert evaluation is a largely manual task, performed by analysts for all but the most ordinary types of alerts.

This has a dramatic effect on analyst efficiency because even where teams are organized in tiers to filter and escalate incidents according to severity, the high volume of alerts in combination with the manual nature of the threat evaluation (triage [2]) process means that skilled and capable analysts frequently spend their time on tasks that do not require their full expertise. Crucially as analyst workload accumulates, there is pressure to make progress by ignoring or insufficiently verifying alerts, carrying the risk of not detecting an incident early enough.

At the same time, security analysts are in short supply. The technical, investigative and cognitive skills necessary for handling security incidents require significant investments in time and training. As a result, the number of skilled and experienced analysts an organization can have at its disposal is limited, making the approach of throwing more "bodies" at the problem unfeasible.

These factors make incident response slow and costly. While many organizations are reluctant to share information on their security incidents, those who do reveal worrisome results. In [4] the authors found that in 54% of the cases examined the time elapsed from the moment of initial compromise to its actual discovery was in the order of months. Another study reported that on average it takes companies 205 days to detect a breach [5]. This is ample time for an adversary to exfiltrate confidential information or infect the infrastructure so pervasively that the cost of recovery is damaging. Crucially, because the volume of security-related information generated from monitoring network and endpoint activity is great (typical data sources include web proxy, firewall, IDS, DHCP and DNS query logs; authentication data; host-based agents; etc.), most organizations find it prohibitively expensive to store it indefinitely. It is not uncommon for organizations to only retain 30-60 days' worth of logs which means that when incidents are detected months after the initial compromise, the response teams may no longer have all the information necessary to evaluate the situation properly.

Worse still, the same survey reports that from the time an incident is detected to its successful containment 38% of the cases required weeks of work [4]. This demonstrates how the various delays introduced at the different stages of incident response can lead to a very slow process overall, during which the organization remains at risk and the costs to the business from the resulting service disruption and recovery operations pile up. As the number of threats facing enterprise networks increases, it becomes clear that the current approach to incident response is not sustainable.

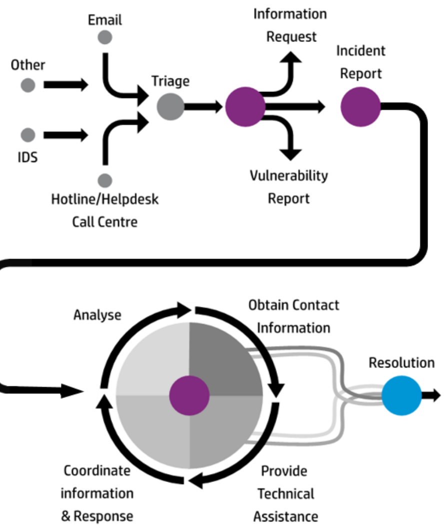

Email
Other
IDS
Hotline/Helpdesk Call Centre
Triage
Information Request
Incident Report
Vulnerability Report
Analyse
Obtain Contact Information
Resolution
Coordinate information & Response
Provide Technical Assistance

Fig. 1. A typical incident resolution process.

A pragmatic approach to security for large enterprise networks, therefore, needs to focus both on minimizing the time between incident occurrence (e.g. infection) and detection, as well as from detection to resolution. While a lot of effort is devoted to the first part of the problem (early detection) through deploying more sensors and developing more capable SIEM and IDS systems, very little has been done to support a systematic capability for rapid response.

Clearly, rapid response necessitates the automation of as many manual processes as possible. Developing automated countermeasures for rapid response however involves numerous challenges. In particular, a key challenge is how to translate the detection of an incident into a set of actionable remediation steps (e.g. blocking network traffic, deploying advanced monitoring or revoking compromised credentials) specific to the threat, that can be automated (entirely or to a significant degree) and orchestrated for immediate execution.

In addition, although the majority of responses involve interaction with a number of network devices such as switches and routers, orchestrating such a response has traditionally been very difficult, as network devices use vendor-specific interfaces and feature-sets requiring changes in the network to be made by experts (i.e. not the security analysts but the networking team), often on a device-by-device basis, slowly and carefully to avoid disruption.

The advent of Software-Defined Networking (SDN) has removed many significant obstacles with regards to the latter issue. SDN allows network devices to present homogeneous management and programmatic access to operators regardless of who manufactured them, what firmware version they run or where in the network they are placed. SDN can thus be leveraged to provide quick and dynamic reactions to threats, once clear actions and rules have been identified. However, the process from identifying new threats to translating them into OpenFlow [6,7] rules as part of a response, in a short period of time, is far from trivial.

This report presents our work towards developing SDN4S ("SDN for Security"): a system that automatically creates incident management workflows to support security analysts in response to an incident alert, handling the selection and implementation of countermeasures and investigation techniques utilizing SDN functions automatically. Central to our approach is the concept of "playbooks" – pre-defined response strategies for each type of threat, outlining the actions that need to be taken, systems that need to be involved and rules that need to be pushed to the SDN-enabled network to tackle the incident, that can be executed immediately, removing the guesswork and streamlining incident response operations.

The rest of the report is organized as follows. In the next section we discuss the concept of playbooks and present the major architectural components of the SDN4S system. Section 3 examines the implementation and deployment details of our prototype. We conclude this paper with a discussion of current and future plans and an evaluation of related academic and commercial work.

II. The SDN4S System

SDN4S was designed to accelerate incident response by automatically creating incident handling workflows, tailored to each threat, to support SOC analysts. The resulting workflows, generated automatically in response to an alert, orchestrate the steps, actions and systems that should be involved in handling the incident, removing bottlenecks and streamlining large parts of the incident management process. As will be discussed later, depending on the incident's severity and the organization's preferences and threat exposure, workflows can be entirely automated or may require an analyst's authorization and input at designated points. This flexibility is intentional. The system is not designed to remove humans from the "driver's seat" but rather to release them from tedious, time-consuming tasks, so that they can focus their experience and resourcefulness on those tasks that matter: "joining the dots", discovering patterns, evaluating information and making decisions.

A. *Playbooks*

The nature and volume of emerging malware threats makes it vital for responses to be flexible and dynamic, i.e. tailored to the specific situation and context at the time of detection. For example, given the same alert indicating infection by a particular strain of malware, the actual threat to the organization (and the required response) may be different if the device involved belongs to the chief executive officer compared to one used by the accounting department, based on the amount of highly-confidential information the former has access to. Furthermore, a given remedy may not be appropriate for every manifestation of a particular threat. Consider a scenario where malware infects a number of different machines, including both servers and desktop computers, While a response where every affected machine is taken offline, wiped and re-imaged will be effective, it may only be necessary for a few key servers to be re-imaged and the rest of the affected machines to have a malware removal tool run on them, especially if the cost from disruption and downtime is factored in.

Organizations thus need to be able to differentiate on how they respond to security incidents in order to be effective in mitigating risks while meeting their business priorities. The challenge is how to provide them with the ability to develop dynamic responses in a way that doesn't introduce further complexity and delays into the incident response process.

SDN4S addresses this problem through the novel use of playbooks. In the world of sports playbooks signify a range of tactics that may be executed by a team during the course of a game based on what the opponent does. In the security incident response context, playbooks similarly describe documents (prepared in advance) which contain instructions on how particular types of incidents must be handled, thus allowing the actual response to be organized and executed immediately once an incident of this type is encountered. Because fundamental elements required to assemble a response workflow (such as key actions to be performed) are already specified in the playbook, precious time that would otherwise be consumed trying to evaluate the best potential course of action is saved. This in turn leads to faster resolution as well as more streamlined and predictable CSIRT operations in general. Importantly, because organizations can maintain separate playbooks for all kinds of different situations, they gain the ability to respond to threats systematically *and* dynamically. We envisage organizations to maintain libraries of playbooks detailing response strategies for various classes of incidents, tailored to their specific business and environment, and to modify, revise and refine these playbooks as the threat landscape evolves and as they learn to develop more fine-grained countermeasures based on them.

Playbooks provide a systematic way to formalize incident response strategies, removing guesswork from even seasoned analysts. They enable strategies to be documented, communicated and shared between teams and individual analysts in an organized fashion, facilitating better evaluation (and development) of techniques and more effective knowledge transfer. Crucially, they provide a means for organizations to express their business and risk mitigation priorities and trade-offs as they relate to a particular situation clearly (by specifying rules, triggers and conditions), relieving analysts from having to interpret them on the spot.

SDN4S playbooks are structured and machine-readable to allow automatic parsing and facilitate easier organization, version control and lifecycle management (design, testing, deployment, evaluation & learning). We use XML to take advantage of the language's ubiquity and the corresponding availability of editors, visualizers and other tools.

SDN4S playbooks consist of two major parts (excluding metadata and annotations): a set of triggers (or conditions) and a corresponding set of executable actions. Triggers specify the symptoms that must be true for a particular playbook to be applicable. These symptoms characterize the threat uniquely and can be as simple as an alert of a specific type coming from a security product to more complex combinations of attributes such as impacted device IP address or VLAN; type of issue; severity classification; user identity; time of day; is traffic encrypted; etc. The use of trigger combinations allows for both coarse- and fine-grained playbooks to be developed as needed.

The second part of a playbook specifies the remediation actions that must be taken once the threat has been characterized. Although in developing SDN4S we are clearly interested in SDN-based mitigations, playbooks can define all sorts of actions and orchestrate a multipart response spanning across multiple systems, as long as that the latter provide interfaces that can be leveraged. For example, a playbook may contain a set of actions which correspond to instructions for the SDN-enabled network and an additional set intended for the Intrusion Prevention System, together creating a two-pronged response. This ability for actions to resolve not only to OpenFlow rules but also to scripted commands is particularly powerful because it enables amongst others further intelligence gathering to be performed as part of developing a response; for example by querying other systems and databases, defining additional probes or passing textual instructions to analysts. By default, actions are executed in their given order. If required however, actions can be executed conditionally, leading to variations to the resulting workflow depending on the state of systems at runtime.

Individual actions may be marked to indicate that they require explicit authorization or input by an analyst before being allowed to execute. As a result, workflows may be designed to execute entirely automatically in response to an alert, or to require an analyst to be directly involved. Automatic and manual steps can be combined in a single playbook as required (e.g. as illustrated in Figure 5 later in the paper). A typical use-case for this is during cases where a step in a response workflow may require privileged access to information or infrastructure, or involve a risky operation. In such cases the playbook is executed up to the point where analyst input is needed and pauses there until the analyst takes control (using the SDN4S UI). Organizations can decide which types of events warrant a fully automated response and which should be escalated to analysts based on factors such as their risk profile, severity of given threat, total amount of threats they encounter, security team workload, infrastructural support, level of situational awareness, etc. The flexibility of the SDN4S playbook implementation means that valuable preparatory work can be carried out automatically before control is deferred to an analyst, allowing them to resolve the incident faster and more effectively.

B. *System Architecture*

SDN4S has a modular architecture to ensure that it can integrate flexibly with multiple systems and control points in the infrastructure, and that its functionality can easily be expanded as needs arise. Figure 2 illustrates the high-level component architecture of SDN4S.

At the heart of the SDN4S architecture lies the Workflow Manager. The Workflow Manager is responsible for assembling the appropriate incident response workflow by combining the instructions found in the corresponding playbook with necessary runtime information (e.g. IP addresses) and passing it to the appropriate systems for execution, including the generation of OpenFlow rules for application by an SDN controller. Playbooks are stored in the Playbook Library which is a data store holding the XML-based playbook documents. A separate Storage Manager manages storage of audit logs (discussed shortly) as well as of any data requiring secure persistent storage. Presentation and user interaction is handled by a web service backend supporting a web-based user interface.

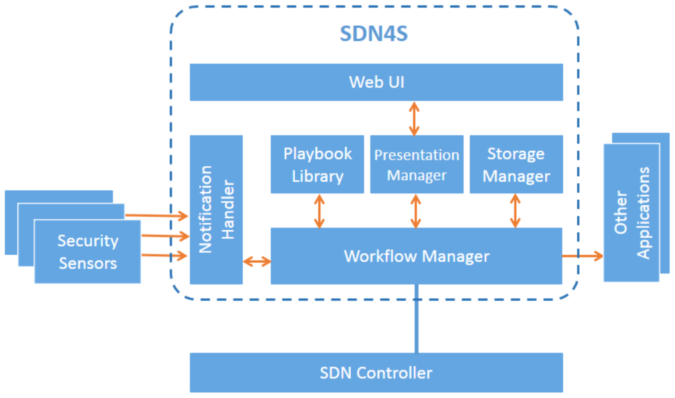


Fig. 2. SDN4S Component Architecture.

SDN4S does not perform incident detection by itself but instead expects to receive alerts of incidents from one or more separate systems. This is by design to ensure that advanced and varied detection methods can be implemented independently from the core SDN4S functionality. SDN4S exposes a set of RESTful [8] interfaces through which systems can “push” alerts to it. This is the preferred method of receiving alerts as it implies a level of awareness of SDN4S by the sources (since they explicitly call the REST interface) which can lead to tighter integration if the latter can provide more tailored contextual information about the alert. This is the approach followed in our prototype implementation discussed in the next section, where SDN4S receives alerts provided by the Big Data for Security (BD4S) system. Alternatively, SDN4S can connect to any alert source (for example a SIEM) via a “connector”: a small piece of software customized to use whatever interface and data model each source may use. Each connector will poll its corresponding source for alerts (or monitor its event stream) and upon finding one will translate the alert information from its native form to the internal SDN4S representation for processing by the SDN4S core engine. Finally, if the source supports pushing of alerts but cannot be configured to support SDN4S natively through its REST interface, a connector can act as a thin “listener”, consuming alerts as they are pushed and simply translating them to conform to the SDN4S internal schema. The Notification Handler component handles alert notification via any of the three approaches outlined above.

The overall SDN4S system operates as follows. Various systems analyze network traffic and client behavior for symptoms of a security incident. When security threats are identified, they are shared with SDN4S among other systems via one of the ways described in the previous paragraph. SDN4S parses the alert and its corresponding metadata and in turn looks up its library of playbooks for a playbook matching the attributes of the alert. It then parses the selected playbook and activates the appropriate incident management workflow, translating the templated actions defined in the playbook into concrete operations targeted to specific devices. For example, if the playbook prescribes a “block all traffic” action for any host matching the conditions defined in it, SDN4S will extract the required information (e.g. IP and/or MAC addresses, etc.) from the alert at run-time and create an OpenFlow rule on-the-fly to be passed to the SDN controller for execution.

Security analysts interact with the system via a custom web-based interface (Figure 4, next section) designed using modern, “responsive” HTML5 technologies to allow easy incorporation into a larger security operations dashboard of the kind commonly used in SOCs, and scale to support a wide range of display sizes. As we will discuss in the next section, analysts are given real-time controls to assess and handle security issues, including the ability to provide authorization for proposed actions or revert mitigations previously applied.

Because of the privileged access to key parts of the infrastructure such as the SDN controller made possible, SDN4S has built-in access controls. Before being allowed to access the SDN4S interface analysts have to provide their security credentials. In addition, SDN4S maintains full audit logs of its operation and analysts’ actions for accountability, compliance, later analysis or simply to calculate CSIRT operations metrics.

III. Prototype Implementation and Deployment

SDN4S is written in Java and acts as an SDN application interfacing with an SDN controller via its northbound interfaces [9]. While developing the prototype implementation, we created two separate versions of SDN4S. A preliminary version was developed to serve as a testing ground for proving the viability of our ideas using the Floodlight SDN controller (version 0.91) [10]. For this version we mostly used a custom-built virtualized network testbed consisting of a few tens of virtual machines (VMs) acting as hosts with different roles (plus a VM-based controller) and Open-vSwitch-based software switches [11]. The network topology of this virtualized testbed was designed to reflect how modern enterprise networks are partitioned into edge, distribution and core segments.

The current version of SDN4S has seen multiple functional improvements based on feedback received on this first prototype by various teams handling IT security and networking operations for a large organization and our own experiments. The current prototype has been modified to use the HPE VAN SDN controller [12]. In addition, we have deployed it on a physical testbed consisting of three HP ProLiant DL360G8 servers carrying out SDN controller and network services (i.e. DNS, DHCP, HTTP, etc) duties and six PC clients acting as hosts. The devices are networked using a topology of two HP ProCurve 3500 and one HP ProCurve 3800 SDN-enabled Ethernet switches.

A. *Big Data for Security*

As discussed in the previous section, SDN4S consumes alerts raised by external systems. In our prototype we leverage the work carried out by the Big Data for Security (BD4S) project [13] developed at HPE Labs. This is a fully working system (currently deployed worldwide within HPE and under customer trial) that can perform big data analytics on huge amounts of security event data to detect yet unknown threats. Focusing on DNS traffic which holds a wealth of security-related information, BD4S performs analytics on huge volumes of DNS packets (of the order of tens of billions of packets per day within HPE). Data is collected at the network layer by packet-capturing endpoint and resolver DNS traffic with FPGA-accelerated network cards. Filtering is performed by dropping whitelisted traffic (approximately 99% of the traffic). Blacklisted and the remaining "grey"-listed traffic is stored in a large columnar database cluster, with the grey-listed traffic being processed algorithmically for previously unknown threats. Analytics are executed on an ongoing basis to identify emerging security threats via trend analysis, computation of threat indicators, and anomaly detection based on historical data. Once anomalous behavior has been identified, an alert is created and sent to the organization's SIEM system, where it can be analyzed like any other security event and qualified as a security incident.

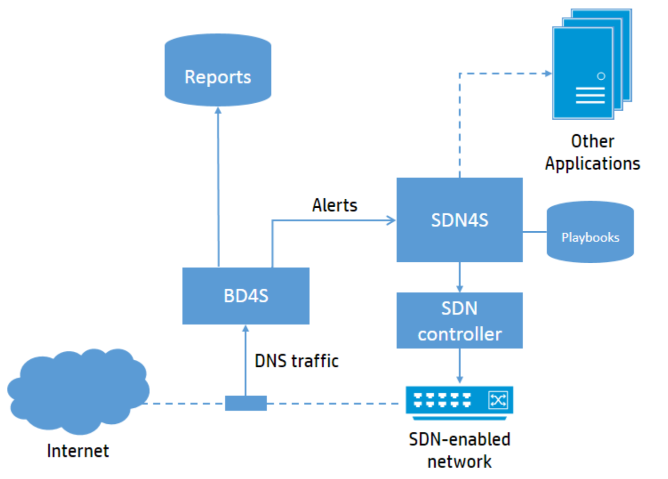


Fig. 3. SDN4S logical deployment diagram.

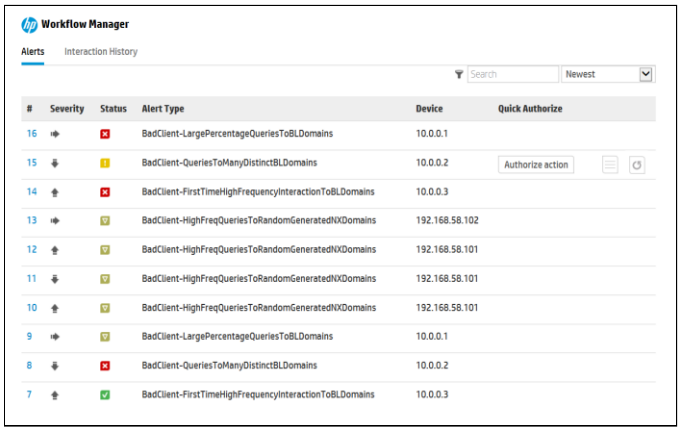
In our deployment BD4S additionally provides its alerts to SDN4S by calling its REST interface (Figure 3). Alerts provided characterize many types of threats, based on symptoms such as queries to blacklisted domain names; abnormal numbers of queries to non-existent domain names; queries to Random Generated Domain names (DGA domains) used by known/unknown malware to communicate with remote Command & Control servers; suspected data exfiltration piggy-backing on the DNS protocol using long and unusual domain names.

B. *User Interface*

Analysts interact with SDN4S via its web-based user interface. The main screen of the UI of the prototype (Figure 4) consists of a list of alerts updated in real time as they are received. Because alerts may come from different sources supporting different native formats, this view provides a homogenized list of key alert attributes extracted from the original message. Specifically, alerts are listed using:

- A unique ID number reference (a simple increment in the prototype), to assist in recording and cross-referencing incidents and cases.
- A severity classification (low/medium/high threat severity), to assist in incident triage. This classification is primarily provided by the alert source but may be re-evaluated by querying additional systems, databases and threat intelligence sources to feed a scoring algorithm.
- A status icon to provide quick, glance-able feedback on the state of the playbook-based response (e.g. whether a playbook executed automatically, analyst attention is required, etc.). Note that in the screenshot of Figure 4 all alerts have triggered associated playbooks, however it is possible that the attributes of a particular alert do not match any of the playbooks available. In that case, the analyst has to process the alert manually, although they can still take advantage of supporting information presented by the SDN4S UI.
- A short textual description of the alert type. This is primarily provided by the alert source but is expected to be in a descriptive, human-readable form. In the screenshot for example, alert types conform to the BD4S naming scheme indicating threats such as devices making "large number of queries to black-listed domains" (line 1) which indicate malware infection with high certainty.
- The affected device IP address.
- A "quick authorize" button to authorize execution of prescribed actions (requiring authorization by an analyst) from the main screen. This is intended to streamline how analysts interact with the system by allowing them to quickly authorize responses when alerts are related to the same incident or no further context is required, without exiting the main view.
- An "alert details" button which loads a new page offering a detailed view of all the information available on the particular alert and affected device, as well as all the steps (both already executed and those pending authorization or action) outlined in the associated playbook. Figure 5 depicts an example of such a screen. The current version of the prototype only displays information provided by the alert source, however plans to augment this information with additional context gathered from AD (Active Directory) servers and other sources is already underway.
- A "roll-back rules" button which reverts all changes made to the SDN network by the OpenFlow rules applied via the associated playbook triggered by the alert. This allows an analyst to override a playbook or remove the applied mitigation if the threat is no longer present.

A separate tab, titled "interaction history" presents a detailed list of all events, alerts and actions that took place during a session, ordered by time, similar to the information recorded in the audit log described in the previous section.



Workflow Manager

Alerts Interaction History

Search Newest

#	Severity	Status	Alert Type	Device	Quick Authorize
16	→	✗	BadClient-LargePercentageQueriesToBLDomains	10.0.0.1	
15	↓	!	BadClient-QueriesToManyDistinctBLDomains	10.0.0.2	Authorize action
14	↑	✗	BadClient-FirstTimeHighFrequencyInteractionToBLDomains	10.0.0.3	
13	→	▽	BadClient-HighFreqQueriesToRandomGeneratedNXDomains	192.168.58.102	
12	↑	▽	BadClient-HighFreqQueriesToRandomGeneratedNXDomains	192.168.58.101	
11	↓	▽	BadClient-HighFreqQueriesToRandomGeneratedNXDomains	192.168.58.101	
10	↑	▽	BadClient-HighFreqQueriesToRandomGeneratedNXDomains	192.168.58.101	
9	→	▽	BadClient-LargePercentageQueriesToBLDomains	10.0.0.1	
8	↓	✗	BadClient-QueriesToManyDistinctBLDomains	10.0.0.2	
7	↑	✓	BadClient-FirstTimeHighFrequencyInteractionToBLDomains	10.0.0.3	

Fig. 4. Main screen of SDN4S user interface.

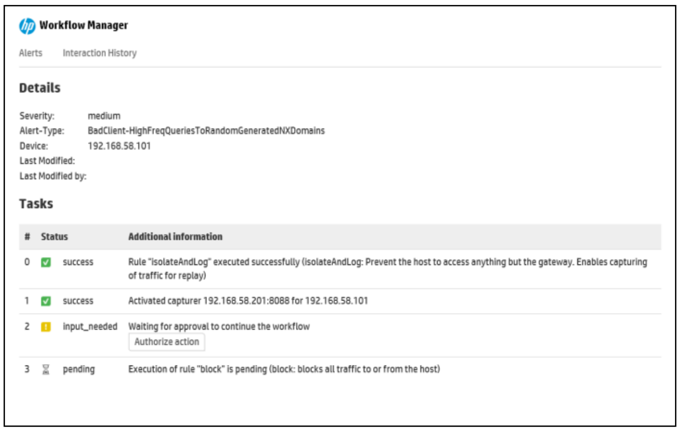
C. *Playbook-driven mitigations*

We have developed an initial set of key playbooks to map incident types into actionable responses. Due to the many variations created by different combinations of trigger conditions and actions we focus this discussion on the types of response strategies they enable:

Block: A strategy centered on preventing a malware infection from spreading from an already infected host to others by preventing network traffic to and from the host. This is a useful building block for many types of responses as it can be used against all types of threats as a "stop the bleeding" measure to buy analysts time while they develop the full response strategy. Crucially, because it is network-driven and not host-driven it does not rely on endpoint (i.e. client-based) security agents which many rootkit-based types of malware can subvert [14]. The playbook produces a flow rule to drop packets that match the criteria of the detected traffic pattern. This resembles traditional IPS (Intrusion Prevention System) or Network Access Control functionality but now can be applied in a more flexible and fine-grained manner since we can flexibly utilize any combination of the OpenFlow 12-tuple [7]. For example, traffic can selectively be blocked to internal network destinations while allowing connections to the global Internet (or specific ports) to pass unimpeded and vice-versa to support different use-cases.

Quarantine: A strategy where devices showing symptoms of malicious behavior are isolated into a separate logical network sandbox (such as a VLAN) where their behavior can be analyzed and managed in a targeted manner without putting the rest of the infrastructure at risk. Like blocking this is a useful (and non-destructive) way to gain time and minimize risk while developing a response plan, especially when the threat symptoms are not conclusive.

Targeted forensics: As discussed in the introduction, the raw volume of network events means that monitoring data can only be collected very coarsely and stored for short periods of time, limiting the amount of available data analysts can base their decision-making on during response planning. SDN4S addresses this problem by enabling selective, fine-grained packet-based monitoring of devices, triggered automatically. Because only the devices involved in an incident need to be observed, storage constraints are relaxed. We have developed a custom-built forensic analysis VM which provides full packet capturing capabilities (presented as a network service configurable via a REST interface) of any packet stream forwarded to it using the pcap [15] library. This allows analysts to use industry-standard tools such as Wireshark [16] and tcpdump [15] to analyze, visualize and process the captured packet streams.



Workflow Manager

Alerts Interaction History

Details

Severity:	medium
Alert-Type:	BadClient-HighFreqQueriesToRandomGeneratedNXDomains
Device:	192.168.58.101
Last Modified:	
Last Modified by:	

Tasks

#	Status		Additional information
0	✓	success	Rule "isolateAndLog" executed successfully (isolateAndLog: Prevent the host to access anything but the gateway. Enables capturing of traffic for replay)
1	✓	success	Activated capturer 192.168.58.201:8088 for 192.168.58.101
2	!	input_needed	Waiting for approval to continue the workflow Authorize action
3	⧗	pending	Execution of rule "block" is pending (block: blocks all traffic to or from the host)

Fig. 5. Detailed view screen for an in-progress workflow.

Clearly, strategies like the above can also be mixed and matched inside a single playbook to stage more elaborate responses, both in fully-automatic and authorization-required modes. To demonstrate this we have created a scenario (and corresponding playbook – cf. Figure 5) where analysts dealing with an infection by a yet unknown strain of malware wish to analyze how it operates by observing the traffic between an infected host and the malware Command and Control (C&C) Server, in order to develop a tailored response and train their security sensors. The playbook workflow opens a new session on the forensic analysis VM by sending a REST request and duplicates all traffic to and from the investigated machine using the appropriate OpenFlow rule, diverting the duplicate stream to the VM for packet-by-packet capture while letting the primary stream reach the external C&C unimpeded. As an added measure however all traffic from the host to internal IPs is blocked to minimize the risk of the malware spreading. A final block rule which requires authorization before being allowed to execute completes the action list. After the analysts are satisfied that they have recorded enough traffic to profile the malware, they can authorize the rule and effectively remove connectivity from the infected host to begin the process of disinfecting it. This scenario demonstrates SDN4S' capability to orchestrate a response combining different systems, as in this playbook multiple SDN rules are combined with the on-demand execution and on-the-fly configuration of the forensic analysis VM.

IV. Current Status and Future Work

We are currently developing new playbooks and mitigation capabilities, especially focusing on the opportunities afforded by combining SDN with software-defined infrastructures such as OpenStack [17]. For example, the creation of "decoys" where an arbitrarily-sized network of highly attractive (and vulnerable)

VMs is created in an effort to distract an attacker from more valuable assets by coupling SDN rules with cloud orchestration such as OpenStack Heat [18].

In addition, we plan to further refine our system architecture and evaluate the system when operating side-by-side with other SDN applications in a production environment. Finally, we plan to evaluate how multiple sources of incident alerts can be utilized intelligently by using a rating algorithm, and the types of probes and queries playbooks can proactively effect in different circumstances to increase the situational awareness of analysts when dealing with a non-fully characterized threat.

V. Related Work

We are not aware of any academic work or commercial product that utilizes SDN functions for supporting security incident response.

HPE Network Protector [19] utilizes SDN to prevent security threats, however it is a preventive tool designed to check DNS queries as they are made in real-time against the RepDV [20] blacklist database and block them accordingly. SDN4S in comparison is a system designed for incident response and although the primary source of alerts for our current prototype also analyzes DNS traffic, SDN4S is designed to be agnostic to how alerts are raised. In all, we consider both systems complementary and expect the two systems to operate side-by-side to cover different security operations use-cases.

Capabilities enabled by SDN4S such as firewalling, network access control, quarantining and packet-capture are also available by more traditional tools such as firewalls and IPS's. The innovation of SDN4S comes with the ability offered to organizations to deploy such controls flexibly and automatically, turning something that was previously infeasible for on-the-fly changes to production networks, to something that is tractable.

Systems like Carbon Black [21] provide an alternative incident response mechanism by having a host-based agent remove network access from the device. This is predicated on an ability to trust a compromised endpoint to be able to enforce policy. In a BYOD-world it is not realistic to expect to have agents installed on all devices, while as mentioned earlier modern malware can employ rootkit techniques to subvert agents. We instead place enforcement controls on the last bastion of the enterprise – the network. From here we can monitor endpoints that are outside of our control/influence and granularly remove their access to the network.

The Netflix FIDO tool for automated incident response [22] was developed for the same reasons we developed SDN4S, namely to accelerate incident response through automation. The existence of FIDO serves as proof that our approach is sound and much needed. Nevertheless, FIDO mostly automates analysis over actions (using playbooks SDN4S can do both), does not employ SDN in any capacity and has a number of hard-coded options for remediation (such as ending a VPN session, or disabling a network port) which are deeply tied to how Netflix's infrastructure is presently put together. As a result these options are not applicable outside Netflix and were not open-sourced.

VI. Conclusions

In this paper we presented SDN4S, a system and solution to minimize the time between incident detection and resolution by using automated countermeasures based on Software Defined Networking (SDN).

SDN4S effectively reduces the lifetime of an individual breach by reducing the time to triage, investigate and remediate security incidents, all the while leveraging systems that improve detection rates via automation. This frees-up analysts to focus their work on refined investigations, and coalesces the collective analysts' activity, allowing optimized resource utilization and therefore an improved security incident response capability.

References

[1] P. Cichonski, T. Millar, T. Grance, K. Scarfone, "Computer security incident handling guide", NIST Special Publication 800-61, 2012.

[2] West-Brown, M. J., Stikvoort, D., Kossakowski, K. P., Killcrece, G., & Ruefle, R., "Handbook for computer security incident response teams (CSIRTs)", No. CMU/SEI-2003-HB-002, Carnegie-Mellon Univ., 2003.

[3] R. Sommer, V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", IEEE Symposium on Security and Privacy 2010, p. 305-316, 2010.

[4] Verizon, "2012 Data Breach Investigations Report", Technical report, 2012.

[5] Mandiant, "M-Trends® 2014: Beyond the Breach", Technical report, 2014.

[6] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., et al., "OpenFlow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, 38(2), 2008.

[7] OpenFlow Switch Consortium, "OpenFlow Switch Specification Version 1.0.0", 2009.

[8] Fielding, R. T., Taylor, R. N., "Principled design of the modern Web architecture", p. 407–416, 2000.

[9] OpenFlow Switch Consortium, "SDN Architecture Overview", ONF TR-504 Version 1.1, 2014.

[10] Project Floodlight, http://www.projectfloodlight.org/floodlight/

[11] Open vSwitch, http://openvswitch.org/

[12] Hewlett Packard, "HP Virtual Application Networks SDN Controller", Technical white paper, 2013.

[13] Big Data for Security (BD4S), Threat Analytics and Visualization Solutions for Big Security Data, https://www.labs.hpe.com/publications/HPE-2016-98

[14] Blunden, B., "The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System", Jones & Bartlett Publishers, 2009.

[15] Tcpdump & libpcap, http://www.tcpdump.org/

[16] Wireshark, https://www.wireshark.org/

[17] OpenStack, http://www.openstack.org/

[18] Heat, https://wiki.openstack.org/wiki/Heat

[19] HP Network Protector, http://www8.hp.com/us/en/products/network-management/product-detail.html?oid=6895435

[20] HP TippingPoint Reputation Digital Vaccine Service (Rep DV), http://h20564.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c03460055&sp4ts.oid=5044950

[21] Carbon Black, https://www.bit9.com/solutions/carbon-black/

[22] Netflix, "Introducing FIDO: Automated Security Incident Response", http://techblog.netflix.com/2015/05/introducing-fido-automated-security.html, 2015.