Securing Networks Using Software Defined Networking: A Survey

Syed Taha Ali[1] Vijay Sivaraman[1] Adam Radford[2] Sanjay Jha[1]

[1] University of New South Wales, Australia
`{taha,vijay,sanjay.jha}@unsw.edu.au`
[2] Cisco Systems, Australia
`aradford@cisco.com`

Technical Report
UNSW-CSE-TR-201330
October 2013

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Software Defined Networking (SDN) is rapidly emerging as a new paradigm for managing and controlling the operation of networks ranging from the data center to core, enterprise and home. The logical centralization of network intelligence presents exciting challenges and opportunities to enhance security in such networks, including new ways to prevent, detect and react to threats, as well as innovative security services and applications that are built upon SDN capabilities. In this paper we undertake a comprehensive survey of recent works that apply SDN to security, and identify promising future directions that can be addressed by such research.

1 Introduction

Computer networks typically consist of hosts interconnected by switches and routers providing data forwarding and routing functionality. These forwarding devices are usually black boxes that run proprietary operating systems and vendor-specific protocols that have to be individually configured in a tedious process in which network operators translate high-level network policies into device-specific low-level commands, manually input using command-line interfaces. The lack of unified network control makes network management challenging, and the painstaking error-prone configuration process is the leading cause of network faults, bugs, and security lapses [1] [18]. Furthermore, due to this inflexibility, network innovation has essentially stagnated, contributing to what some term the Internet "ossification" [27] phenomenon.

The recently emergent Software Defined Networking (SDN) paradigm [57] addresses this challenge by separating the packet forwarding functionality of the forwarding devices, i.e. the *data plane* from the control element, the *control plane*, that runs as logically centralized software. This decoupling enables a radical new network architecture, depicted in Fig. 1.1: switches in the network are reduced to basic packet forwarding devices that are remotely managed by a centralized software-based 'controller' entity. The controller programs switches remotely using the OpenFlow protocol, a standard and open interface. This abstraction is analogous to an *operating system* [33] in which the controller acts as the OS kernel which abstracts the forwarding hardware of the network and presents application developers with a unified programmable interface on which to deploy software and applications.

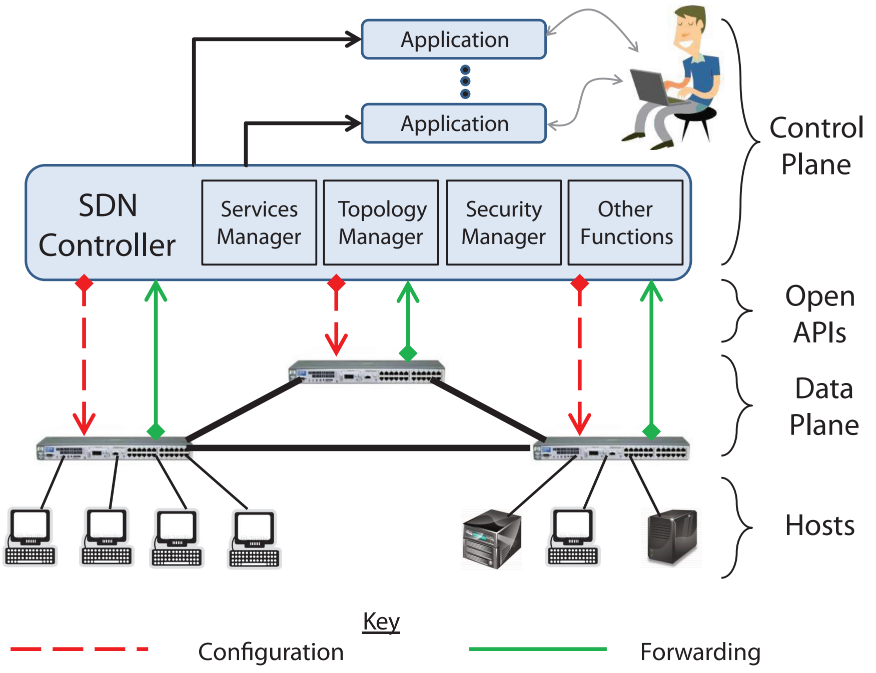

Figure 1.1: SDN Architecture

This reorientation of control and forwarding elements in the network enables considerably simpler, faster, and more effective network management. Network operators no longer need to enforce complex policies manually on individual devices, instead they specify high-level declarative policies for the network as a whole which the controller translates and installs in switches in the form of localized flow rules. In addition, the controller polls flow statistics from network devices periodically, thereby compiling a centralized real-time view of network state. This state can be exposed via open APIs, allowing software application developers to automate the control process, and enabling dynamic and efficient network management. Examples of innovative network management applications include dynamic load balancing [79], advanced threat mitigation [74], and virtual machine mobility [51].

SDN has captured the imagination of the research community and industry alike and the movement towards adoption has gained considerable momentum. Universities are actively building SDN deployments and testbeds [5], [83]. Google has already deployed SDN in-house for datacenter backbone traffic and reports an unprecedented network utilization of up to 95% [42]. There is significant interest from industry vendors: major commercial switch vendors including Cisco, IBM, Hewlett-Packard, Dell, Juniper Networks, Brocade Communications and others have announced intent to support or have already launched switching products that support the OpenFlow protocol. Industry and vendors have also launched collaborative efforts such as the Open Networking Foundation (ONF) [3] and the OpenDaylight project [4] to promote SDN standardization and innovation, and the IETF and IRTF have also set up SDN working groups. Market analysts agree that the SDN market will grow at a phenomenal rate: IDC predicts it will be worth $3.7 billion by 2016 [44], whereas Plexxi estimates growth to be an order of magnitude higher, and expected to surpass $35 billion by 2018 [67].

The ability to view network state in real-time, and programmatically control network behavior, opens up exciting possibilities for network security. Security has assumed critical importance in recent years due to a number of reasons. As adoption rates for disparate networked devices such as mobile phones, tablets, and sensors skyrocket, networks face a ‘crisis of trust’ [76] and reports indicate that security threats continue to grow and evolve very rapidly [20, 16]. Furthermore, critical and aging infrastructure, such as automation systems, factory equipment, and traffic controls, are being brought online without adequate security and safeguards [75], amid growing fears of ‘irrational’ hackers [28]. In this paper we examine how SDN offers new ways to tackle these important concerns.

We categorize current SDN-based security research into two branches, research geared towards protecting the network, and providing security as a service. The first direction, reviewed in Section 2, deals with **security configuration** and **threat detection, remediation and verification** using SDN. The consolidation of policies at the central controller enhances consistency of configuration, helping prevention of attacks. The centralization of network state makes it easier to detect intrusions and anomalies, and to react in an agile and coherent way to isolate or neutralize the attack. As examples, a Denial of Service attack can be inferred from current network state, and a threat mitigation application may dynamically reprogram switches at the network perimeter to drop malicious traffic flows; a malware outbreak in one section of the network can be contained by instructing select switches to restrict traffic flows to that section; and a user on an infected machine can automatically be routed to a web-server issuing a quarantine notification. An analogy may be drawn to the 'immune system response' [38] where a centralized intelligence collects information from network elements and dispatches the appropriate response to the infected part.

The second branch of SDN-related security research, reviewed in Section 3, develops innovative security capabilities that can be instantiated on-the-fly, thereby offering **security as a service**. For example, user identity such as IP address may be anonymized, affording privacy from targeted advertising or government surveillance; sensitive data may be offloaded securely within an enterprise network or across organizational boundaries; and management of security threats may be out-tasked to specialized third parties. These security capabilities need not be always on, they may be selectively invoked on a need basis for specific traffic flows, thereby permitting an elastic cost model for the value-add services.

We believe that research in SDN-based security is still in its early days, and there are several areas in which SDN can play a pivotal role. In Section 4 we discuss some promising new directions, including using SDN to federate heterogenous networks, couple overlay networks with hardware, and for service chaining using network function virtualization.

2 Protecting the Network

In this section we survey the basics of security configuration using SDN, and discuss new techniques for threat detection, remediation, and network verification.

2.1 Security Configuration using SDN

Control Plane Centralization

The original vision for software defined network security management is spelt out by Casado et. al in **SANE** (Secure Architecture for Network Enterprise) [15], a clean-slate security solution for enterprise networks. Enterprises today face a barrage of ever-evolving security threats and have little choice but to rely on a combination of security solutions that are complicated, distributed, and limited in scope. Security policies are typically implemented as complex, topology-dependent access control lists. Trust is distributed across multiple components, such as switches, DNS servers, authentication services (such as Kerebos [65] and RADIUS [71]) and each of these individual components need to be protected in turn. If a network element is compromised, an attacker may be able to identify vulnerabilities and obtain sensitive information about the network itself, such as the topology, the location of critical servers, etc. Furthermore, security enforcement at higher layers may be undermined by unsecured access at the lower layers.

In SANE, all connectivity in the network is mediated by a single protection layer that is overseen by a logically centralized controller in charge of all routing and access control decisions. High-level declarative policies can thus be specified in terms of services and principals that are independent of topology, e.g. *Alice* *can contact service ftp* or *users in group bobs-friends can access bob's streaming-audio server*, and the controller accordingly configures encrypted access routes across switches and routers. Security is enforced at the link layer and cannot be undermined by lower layers. The controller is the sole trusted entity in the network and it restricts network access for unauthorized parties.

Deploying SANE, however, requires significant upgrade to the entire network infrastructure and modifications at the hosts. To ease deployment issues specifically, the authors extend their original work and propose **Ethane** [14]. Ethane retains SANE's primary vision of a centralized controller and introduces the additional element of Ethane Switches, which consist of basic forwarding hardware with minimal memory able to track flows in-progress. Ethane Switches can be deployed alongside Ethernet switches in the enterprise, and the controller ensures seamless traffic flow across them, thereby allowing enterprises the option to upgrade their networks incrementally. Furthermore, Ethane does not require modification at the hosts.

Flexible Policies

Software defined networking also allows network administrators to define intelligent and dynamic policies that strike a cost-effective balance between users' convenience and network protection. A popular example is that of Ballarat Grammar [50], a school in Victoria, Australia. The school, catering to over 1400 students, sought a campus-wide network security solution to facilitate WiFi access for personal devices, such as laptops and tablets, and would not be restrictive on students and staff. The school installed OpenFlow firmware on its switches and used Hewlett-Packard's Sentinel Security SDN application [37] to route DNS queries through an intrusion prevention system. This approach effectively filters for malware on user devices without having to install and manage specialized software on individual users' devices.

Administrators at Ballarat Grammar report a further advantage: previously when a machine would be infected, a user would report it to the IT department which would initiate diagnosis and repair, a process spanning several days to weeks. After deploying SDN, however, infections and malicious activities are detected and logged in detail by the Sentinel application, significantly aiding with trouble-shooting and reducing downtime.

2.2 Threat Detection

The SDN paradigm offers a new level of visibility into the network which is ideally suited for traffic monitoring applications. The controller can program forwarding devices in the network to conduct fine-grained packet inspection on traffic passing through the devices. These statistics, periodically collected by the controller, afford a centralized real-time view of network state which is exposed via open APIs, allowing for automation. Developers can write applications utilizing data mining and machine learning techniques to enable rapid intelligent identification of threats.

There is also the advantage of flexibility and scale: Microsoft has revealed that it uses a homegrown SDN solution to capture and analyze the huge volumes of traffic in its Internet-facing and cloud services data centers [54]. Data centers typically consist of thousands of 10 Gigabit Ethernet links, and traditional packet capture mechanisms such as port mirroring and SPAN, which require an immense number of physical ports, are infeasible from a scale and cost perspective. On SDN-compatible switches, virtual ports can be defined with ease for packet monitoring purposes. Furthermore, network operators can define software policies to create service chains, diverting flows through multiple analysis and inspection points. This obviates the need to insert dedicated middleboxes at traffic chokepoints in the physical network.

Denial-of-Service Attack Detection

Security solutions provider Radware has recently developed **DefenseFlowTM** [61], the first commercial SDN application that addresses denial-of-service (DoS) attacks. Radware has furthermore contributed a simplified open source version of DefenseFlow, Defense4All, to the OpenDaylight project [55].

DefenseFlow directs the network controller to collect specific flow statistics from forwarding devices in the network at a per second resolution. The application measures baseline traffic flows and then monitors for patterns suggestive of a DoS attack. In the event that a threat is detected, a traffic diversion mechanism programmatically redirects suspicious traffic to a dedicated scrubbing center (running Radware's DefensePro network behavioral analysis system) for detailed traffic inspection, signature analysis, and threat neutralization.

Traffic Anomaly Detection

SDN-enabled distributed traffic inspection functionality also has application to anomaly detection solutions. Anomaly detection mechanisms running on Internet core routers cannot process adequately the high volumes of traffic flowing through at line rates, and, additionally, these mechanisms generate a large number of false positives, which cannot be dealt with practically in the network core.

Mehdi et al. [58] make the novel suggestion that the user home gateway device is ideally positioned to run anomaly detection mechanisms. The advantage of the home gateway is that residential data rates are low enough to enable anomaly detection at line rate, false positives can be screened more efficiently, and specialized security policies can be managed on the device by a remote controller.

The authors implement four anomaly detection algorithms for the NOX SDN controller (which they have made publicly available). These include the Threshold Random Walk with Credit Based Rate Limiting (TRW-CB), Rate Limiting, Maximum Entropy Detector, and NETAD algorithms. Experiments are undertaken with real-world traffic datasets, into which the authors inject portscans and DoS attacks at varying rates, and perform anomaly detection at three points in the network: the edge router of an ISP, a home network router, and a switch in a small office. Results indicate that the algorithms are unable to satisfactorily identify anomalies at the ISP level but prove highly accurate running at the home and small office level.

2.3 Threat Remediation

In traditional networks, the only possible response to a threat has been to drop offending traffic. SDN, however, with on-the-fly programmatic capabilities, makes possible a richer variety of dynamic responses, including emergency alarms, dynamic quarantine solutions, traffic redirection for forensics, and entrapment mechanisms such as tarpits and honeypots.

FRESCO [74], proposed by Shin et. al, is an application development framework facilitating design of sophisticated threat detection and mitigation modules. FRESCO provides a scripting API and basic reusable modules, which can be assigned relevant parameters and stitched together into a desired security configuration. At compilation, these modules produce flow rules which are overseen by FortNOX [68], a specialized security enforcement kernel which is embedded in the network controller.

The authors provide two case studies to demonstrate the power and range of FRESCO: first they build Reflector Net, an application to detect and entrap malicious scanners. If an attacker initiates a large number of failed TCP connections, the ScanDetector module is triggered, prompting the ActionHandler module to redirect the traffic to a remote honeypot. The attacker therefore receives valid responses from the honeypot machine, under the impression that it is still communicating with the original target. In the second example, the authors demonstrate how FRESCO can be integrated with legacy security applications: monitoring tools such as BotHunter [40], in the event that they detect a threat, can invoke security applications written in FRESCO script to quarantine infected hosts on the network.

The FRESCO Application Layer prototype is implemented in Python and operates as an OpenFlow application on NOX. The kernel, FortNOX, is implemented directly into NOX as a native C++ extension. However, the architecture and methodology can easily be ported to other SDN designs and controllers. Modules are implemented as Python objects. The research team has built 16 commonly used modules (including a FRESCO Scan Deflector, an adapted version of the Botminer [32] application, and a P2P malware detection service) with plans to build more and release to the research community.

2.4 Network Verification

A popular area of research is the use of automated techniques to verify network consistency in SDNs. Human operators are prone to make errors: security professionals attending the DEF CON 18 conference recently reported encountering poorly configured networks "more than three quarters of the time" and they were in strong agreement that badly configured networks are the main cause of network breaches [1]. A Gartner study predicts that in the period 2010 to 2015, 80% of network outages impacting mission-critical services will be due to 'people and process issues', and more than 50% of these stemming from configuration modifications and updates and hand-off problems [18].

In software defined networks, such problems may be encountered when network controllers are shared by different users or applications, or multiple controllers operate in the same domain, leading to conflicting rules, violation of policy, or network faults, such as loops, black holes, access control violations, etc. Malicious parties may even bypass security policies by defining strategic flow rules to re-label and redirect traffic. Furthermore, in the case of large networks potentially comprising hundreds of switches, where multiple applications are able to program the network and SDN controllers have the capability to install approximately 50k new flows per second [80], there needs to be quick and efficient mechanisms to ensure security compliance, fault tolerance and fast failover.

Formal reasoning techniques are a powerful tool in this regard. The SDN paradigm simplifies the traditional network in two very important ways: first the network no longer consists of disparate elements running proprietary protocols but instead comprises uniform switching hardware with standard functionality and interfaces, communicating using a single open protocol. Second, network control is no longer distributed over several elements, and is now centralized in a single controller device. The state and behavior of the network therefore is the logical outcome of configuration commands dispatched by the controller, and these can easily be modeled using formal techniques. This allows administrators to fault-check networks, verify network properties, and build in failsafe mechanisms. Formal techniques are already being applied in designing 'machine-verified' network controllers [34], programming languages for software-defined networks (such as Frenetic [29]), and innovative abstractions with verifiable security properties such as isolated network *slices* [36] and security monitoring routing protocols [10].

The earliest work in this area, **FlowChecker** [6] is a property-based verifier solution to identify misconfiguration within the network. FlowChecker encodes switch flow-table configuration using Binary Decision Diagrams to model a state machine describing the behavior of OpenFlow switches in the network. FlowChecker then uses the model checker technique to validate correctness of the interconnected network. Network administrators can also use this solution to analyze the impact of new applications on the network prior to installation.

NICE [13] uses automated model checking and symbolic execution to identify bugs in SDN applications. The network operator inputs details about the controller, network topology specification, number of switches, hosts, etc. NICE tests applications on this schematic by automatically generating specific traffic flows that study the network for a variety of different events, and identify property violations. The authors prototype NICE using Python for the NOX controller and test three real applications (a MAC-learning switch, a Web server load-balancer, and energy efficient traffic engineering) and uncover several bugs.

VeriFlow [46] is a solution which verifies network correctness in real-time as the network evolves. VeriFlow is situated as a layer between the controller and the switches and checks the validity of network invariants whenever a new forwarding rule is installed. VeriFlow divides the network into a set of equivalence classes on the basis of existing rules. Packets falling into a class undergo the same forwarding decisions throughout the network. When a new rule is to be introduced, the classes that will be altered by such a rule are located and network invariants are verified within those classes. VeriFlow maintains forwarding graphs for the equivalence classes and traverses them to query the invariants. Each flow modification is thus verified in real-time before it is implemented. The VeriFlow prototype implementation, run using a NOX controller managing a simulated OpenFlow network on Mininet, verifies that VeriFlow has minimal impact on network performance and can verify network-wide invariants in near real-time, within hundreds of microseconds.

FortNOX [68] is a NOX-based security mediation service that is directly integrated in the NOX controller, and is able to identify conflicting flow rules in real time. This is done by converting all rules to a representation the authors refer to as 'alias reduced rules' which can then be easily validated. In case of conflicting flow rules, the choice of flow rule is made depending on the relative level of security authorization of the requesting party. In case the flow rule is not implemented, the service returns an error message to the application.

FLOVER [77] is another solution to verify compliance of dynamically assigned rules with invariant security policy in real time. FLOVER translates security policies into a set of assertions (referred to as Non-bypass properties) which can be processed and verified using an SMT (Satisfiability Modulo Theories) solver. Experimental results (using a simulated OpenFlow network on Mininet with a NOX controller) indicate that FLOVER can detect coverage and modify violations of up to 200 rules in under 131 ms.

Apart from real time network verification, there is also a need for fast failover mechanisms to cope with network disruptions. Typically, if a network link fails in an OpenFlow network, the switch informs the controller which programs new flow rules for all the affected switches. This process takes some time [73] as the controller will have to compute alternate paths, e.g. run spanning tree protocol, verify that new paths do not violate policies, etc. and during this period running traffic in the network may suffer from inconsistent flow and policy violations.

Newer versions of OpenFlow, however, enable multiple forwarding behaviors to be defined for the same switch, differentiated on the basis of switch state and specific parameters, such as whether a link is up or down. Reitblatt et. al [69] capitalize on this feature to formalize network correctness and fast failover in a high level declarative application programming language called **FatTire** (Fault Tolerating Regular Expressions). With FatTire, programmers do not need to explicitly program failover forwarding rules, they simply specify policies and the controller precomputes appropriate backup paths and populates the switches with flows and backup flow paths. Now, in case a link or device in the network fails, the controller no longer needs to intervene, the network elements switch automatically to the backup rules.

A FatTire program has three policy components: a security policy (e.g. *all SSH traffic must traverse the IDS*), a fault tolerant policy (e.g. *forwarding must be resilient for a single-link failure*), and a routing policy (e.g. *traffic from the gateway to the switch can be forwarded using select paths*). The system then computes all possible paths for these specifications, and performs the intersection operation, to identify those which fulfil all criteria. These policies are then translated into OpenFlow rules using a modified version of the NetCore compiler [62]. The authors have prototyped their solution which takes as input a FatTire application and a network topology, and outputs policies for individual elements in the network.

3 SDN Security as a Service

In this section we examine how the SDN paradigm facilitates additional network security measures which go beyond network protection, enabling services such as anonymization, enhanced trust, and remote management.

3.1 Anonymization

In [60], Mendonca et. al present an SDN-based anonymization service to counter IP-based profiling on the Internet. Sophisticated data mining systems today are able to piece together significant detail about Web users from their IP addresses, attributes such as location, Internet usage patterns and possibly even their identity. Static IP address assignment also renders a user vulnerable to online censorship, active attacks, and breach of data. Existing approaches to reclaim anonymity, such as Onion routing overlays (e.g. ToR and Tarzan) are inefficient and have significant latency issues.

The authors propose **AnonyFlow**, a solution where Internet users offload trust to their primary ISPs which assign temporary IP addresses and disposable flow-based identifiers to user traffic exiting their domains. This is similar to the concept of NAT: third parties on the Internet no longer have visibility in the ISP network and are therefore unable to correlate user traffic to specific IP addresses with which they build composite user profiles. The authors present this as an additional service an ISP may choose to provide, similar to the caller-ID blocking service which withholds customers' phone numbers and call information from call recipients.

Previous attempts at such anonymization solutions required specialized gateways to track user state and traffic flows. The SDN paradigm dramatically simplifies this deployment: the controller can coordinate the implementation of custom routing policies across multiple switches in the network on an on-the-fly basis. The anonymization function is performed by the switches at line speed. Testbed results using commercial OpenFlow-enabled switches reveal that AnonyFlow causes near-negligible impact on throughput.

A similar approach is taken by Jafarian et al. [41] to anonymize network hosts to protect from online adversaries. Adversaries typically use stealth scanning tools to remotely probe IP addresses at random in networks to identify targets. When a host responds, attackers can probe it further to identify vulnerabilities and launch specific attacks. A typical solution is to dynamically modify host IP addresses using NAT or DHCP but the address assignment may be infrequent and traceable, and inconvenient in that active connections are disrupted during an address change. The authors therefore propose **OpenFlow Random Host Mutation (OF-RHM)**, a mechanism to *mutate* IP addresses of network hosts randomly and frequently in a fully transparent manner. In this case, the controller assigns each host a random and temporary *virtual* IP that is translated to/from the host's *real* unchanging IP address. External parties can access the hosts using their virtual IPs which are circulated via DNS. Access to the host using the real IP is restricted solely to authorized parties.

OpenFlow-enabled network switches perform the translation between real and virtual IPs and the controller coordinates the mutation process across the entire network. Virtual IP addresses are picked randomly from a pool of unused addresses in the network. Host IP mutation is set to be rapid and unpredictable for maximum efficacy, and even in a limited, fragmented address space, each host should mutate at required rate such that no IP address is reused for a long period. The authors treat these requirements as a constraint satisfaction problem, solved using Satisfiability Modulo Theories (SMT) solvers. They propose two mutation strategies, Blind Mutation, where virtual IPs are chosen from the address space with uniform probabilty, and Weighted Mutation, where the selection is weighted based on previous usage of the particular IP address. The authors implement OF-RHM in Mininet with multiple NOX controllers and theoretical results indicate that OF-RHM invalidates up to 99% of the information gathered by remote scanners, saving up to 90% of hosts from sophisticated zero-day worms.

3.2 Out-tasking Network Security Management

Small business and home networks are particular targets for online attack. The reasons are primarily lack of specialized security protection (e.g. dedicated firewalls, IPS systems, security expertise) due to the cost factor, a homogenous and predictable network architecture, and an erroneous belief in security through insignificance [39]. Kindsight Security Labs report that 10% of home networks are infected with malware in the second quarter of 2013 [47]. Small businesses are particular targets, they are the most victimized in the category of businesses; 31% of recorded cyberattacks in 2012 targeted businesses with fewer than 250 employees [43]. The Federation of Small Businesses, the UK's leading business organization, comprising some 200,000 small UK businesses, states that cybercrime currently costs its members up to 800 million pounds per year [70].

Feamster [25] points out that the SDN paradigm enables users to outsource their network security to professional third parties who possess operational expertise as well as a broader view of the network. There are already preliminary moves towards such an arrangement: basic ADSL modems can be easily configured to to use third party DNS services such as Google DNS and OpenDNS which provide protection from a range of malware, typo protection, phishing and optional content filtering. Software-defined networking enables a complete transition and give full control to a third party service. Research trials are already underway [52] where ISPs are installing Lithium OpenFlow controllers to remotely manage home networks on behalf of residential customers.

The basic model for residential users is as follows: programmable network access points in the home periodically dispatch traffic statistics to the (remotely based) controller which runs specialized algorithms to detect spam filtering and botnet detection, and implements the relevant protection policies (filtering rules, blacklists, etc.). The author identifies open research challenges in this scenario: devising a methodology for collecting statistics that strikes the appropriate balance between the scale of the data collected, efficacy of threat detection and maintaining user privacy. Suggested solutions include intelligently sampling the data depending on specific deployment scenarios and making the process dynamic (i.e. the controller could have the ability to demand finer-grained statistics if the need arises). The issue of collecting appropriate statistics on flows for security monitoring is explored in greater detail in [84], and specifically in the context of home networks in [12] and for large networks in [10].

To protect user privacy, the author proposes that ISPs or home users obfuscate IP addresses in flow statistics and anomaly detection algorithms work (on aggregate data) at the level of IP prefixes rather than specific IP addresses. Furthermore, the user may locally deploy privacy preserving algorithms to screen sensitive data before the results are uploaded to a central server.

3.3 BYOD and Secure Data Offloading

Software defined networking can also be leveraged to secure data offloading from mobile and handheld devices. Smartphones and tablets greatly enhance productivity, but sensitive data may need to be offloaded from the device on to the network for further processing, sharing, or archival. Dictaphone audio recordings may be offloaded to remote servers for software-based transcription, and patient data may be collected from bodyworn medical sensing devices for cloud-based data mining. There is also a trend among business and corporate employees to use their personally owned handheld devices to access corporate networks, creating a wealth of security problems. In such situations, SDN can dynamically slice and allocate network resources in a secure and elastic way as per the specific requirements of offload applications.

Gember et al. [30] identify in more detail the hurdles facing secure data offloading: offloading may require energy-intensive processing on the mobile device to capture execution state and the data transfer may incur problematic latency issues. Additionally, offloading must permit differentiated security settings, according varied access privileges and protections to different classes of data. Offloading, furthermore, must not rely on dedicated resources (which may cause bottlenecks in the system or single points of failure), and it must be scalable (multiple applications may vie to offload data with different objectives and policies).

To meet these needs, Gember et al. propose **Enterprise-centric Offloading System (ECOS)**, an enterprise-wide solution enabling mobile applications to offload data in a tightly controlled environment as per the privacy, performance, and energy constraints of users, and process the data by opportunistically leveraging diverse compute resources. The network controller negotiates the requirements of the mobile application, e.g. is encryption feasible with current energy resources, are trustworthy compute resources currently available, what level of security is required, etc. Applications can choose if they want energy savings or latency improvements and the controller leverages idle resources from the enterprise pool. Data is also graded into distinct categories, user-private, enterprise-private, and no-private, enabling the controller to implement differentiated service policies for each type. Since SDN manages flows in real-time, the mobile device can also move within the network during the offloading process.

ECOS is prototyped using OpenFlow and Android. Experimental results indicate that latency improves by as much as 94%, energy savings up to 47%, and as much as 98% reduction in execution state that applications need to communicate.

4 Future Directions

In this section we discuss potential challenges and future security-oriented applications for software defined networking.

4.1 Federating Heterogenous Networks

The SDN programmable networks vision is already being applied to unify network management in WiFi networks [63] [78], to provide programmable interfaces across the wireless stack [11], for accessing services in IEEE 802.15.4 networks [21], and in coordinating services across heterogenous networks [59]. It is anticipated that SDN will also be a key driver in the emerging Internet of Things paradigm [82], where the centralized control element and standardized protocols facilitate the process of federating disparate devices such as mobile phones, smart TVs, computers, household appliances, sensor devices, healthcare monitors, etc. and enable implementation of federated policies on top. How to manage privacy and trust in a seamless and efficient manner across heterogenous networks and multiple devices is a critical question.

There is already a marked trend in this direction in home networks research: networking appliances and devices in homes and buildings is now practical [23] [22] [9], and there is a critical need to assure the privacy and security of residents [8]. The model for deployment of home networks is similar to SDN in that there is a centralized intelligence: certain research and industry efforts [81] [2] recommend enhancing the capabilities of the residential Internet home gateway device to serve as the controller entity and orchestrate operations between high-level user-defined applications and networked devices.

There is considerable potential for new research in the design and implementation of security policies for such scenarios. Smaller devices, such as actuators and sensors, generally possess limited compute and battery resources, and cannot run resource-intensive cryptographic protocols. There is therefore an urgent need for mechanisms which translate security privileges across domain boundaries and contribute to enforcing a *uniform* federated security policy in a seamless and efficient manner. We expect the controller will play a key role in this regard. Existing research in wireless sensor networks may prove useful: an illuminating example is that of Sizzle [35] which implements a minimal footprint HTTPS stack between two endpoints, a Web browser and a miniaturized wireless device. With Sizzle, a user can control resource-constrained wireless devices (such as sensors, thermostats, etc.) from a browser window with end-to-end SSL protection. As we noted earlier, standard SSL operations are not practical for small devices. In this case, a basestation acts as a gateway device between the Internet and the wireless sensor network, and amortizes the high cost of an SSL handshake across multiple data transfers.

4.2 Coupling Overlays and Underlays

With advancement in virtualization technologies, overlay networks are now viable. Successful examples include Midokura MidoNet [53], Nicira NVP [66], IBM's Distributed Overlay Virtual Ethernet (DOVE) [19] and PLUMgrid [56]. There is therefore strong potential for deploying overlay networks to provide specialized security services such as enhanced user privacy, anonymization, and network route management. Overlay networks enable finer customization, differentiation in services, and remote management, but generally suffer from performance issues due to the loose coupling between the virtual overlay and the underlying physical hardware. Tor is an example of an anonymizing overlay that suffers from severe latency issues (a usability study noted that DNS requests on ToR were 40 times slower [24]).

Overlay networks can use native SDN protocols and open APIs to fully harness the high speed and reliability of the underlying physical network. Using SDN, highly efficient overlay networks can be set up with just a few clicks and configured with specific security properties. For instance, overlays can be designed to enable secure channels for online services for routine everyday tasks such as banking and online shopping. Overlays could also provide routing control as a service to counter DoS attacks [45] or restricting traffic flow to trusted regions in the network [48]. More estoric possibilities include bypassing censorship (such as the Infranet project [26]), or constructing specialized overlays for sensitive functions such as anonymization [49], military operations, smartgrid control, electronic voting, etc.

4.3 Beyond OpenFlow and Network Functions Virtualization

Currently we are also witnessing a move towards instantiating sophisticated compute capabilities in the network itself. This trend goes a step beyond the OpenFlow SDN vision (exemplified in the work we have covered thus far) in which switches and routers are considered minimal data forwarding devices. In contrast, **Switchblade** [7] uses programmable FPGA hardware to deploy extra features 'on-the'fly' in forwarding elements, providing functions such as customized protocol processing, path splicing, etc. Similarly, the **Cisco onePK** [17] platform, motivating Cisco's Internet of Everything roadmap, seeks to build greater functionality into the forwarding path, delivering a range of functions such as encryption, transcoding, and deep packet inspection, and release APIs which give developers fine-grained control over these processes.

Narayanan et. al [64] suggest an application extensibility framework for deploying middlebox functionality (such as encryption) on programmable switches which is compatible with the OpenFlow protocol. Their solution abstracts the packet processing modules on the switch and creates a virtual port on the switch. Network operators desiring to use the processing functionality can use OpenFlow rules to route traffic to the new port.

A complementary trend is that of **Network Functions Virtualization (NFV)**: specialized network middleboxes (such as firewalls, encoders/decoders, DMZs, deep packet inspection units) also suffer from lack of innovation in that they are typically closed black box devices running proprietary software. Researchers have proposed that specialized middleboxes be defined entirely as virtualized software modules and managed via standardized and open APIs [31] [72]. This brings in the benefits of reducing expenditures in purchasing and maintaining customized hardware, and time-to-market for new services is accelerated. However, the real value of NFV lies in meeting economics of scaling demand. With the traditional model, growing demand for services requires purchase and installation of new hardware which takes time and effort. With NFV, network services are elastic, and new resources can be allocated to meet demand in minutes.

SDN is essential to interconnect these virtualized network functions in a dynamic and transparent manner. Developers can write powerful applications by stitching virtualized function modules in desired service chain configurations, and use SDN protocols to optimize traffic flows along the chain and maintain end-to-end QoS and policy control. Carriers can use NFV/SDN in this manner to manage service flows more accurately and efficiently.

Open research questions in this context include how to architect and manage these middleboxes satisfactorily, to address the inevitable increase in latency when hardware functionality is coded in software, and how to best distribute functions across the network, i.e. increase 'in-network' capabilities for security functionality such as firewalls, network caching, and DRM-management in an efficient scalable manner.

5 Conclusion

Research in software-defined networking is still in its early stages, and we consider it a healthy sign that there is already significant work being done to develop innovative new security solutions and applications for these networks.

In this paper, we have undertaken a comprehensive review of security-oriented research in software-defined networks. We have classified current work in two main streams: *threat detection, remediation and network correctness* which simplify and enhance security of programmable networks, and *security as a service*, which offers new innovative security functionality to users, such as anonymity and specialized network management.

Furthermore, we discuss possible challenges and future directions for security in SDN: these include orchestrating security policies across heterogenous networks, customizing overlay networks to provide secure environments, and extending the OpenFlow paradigm with customized hardware and network functions virtualization and building a richer set of features in the forwarding path.

Bibliography

[1] ”Hacking Habits” Survey Cites Misconfigured Networks As The Main Cause Of Breaches. 31 August, 2010. `http://www.tufin.com/about-us/news-and-media/press-releases/2010/august-31,-2010/`.

[2] *HGi: The Home Gateway Initiative*, Retrieved 14 June, 2013. `http://www.homegatewayinitiative.org/`.

[3] *Open Network Foundation*, Retrieved 14 June, 2013. `https://www.opennetworking.org/`.

[4] *OpenDaylight Project*, Retrieved 14 June, 2013. `http://www.opendaylight.org/`.

[5] *OpenFlow Network Research Center*, Retrieved 14 June, 2013. `http://onrc.stanford.edu/`.

[6] E. Al-Shaer and S. Al-Haj. FlowChecker: Configuration Analysis and Verification of Federated OpenFlow Infrastructures. In *Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration*, pages 37–44. ACM, 2010.

[7] M. B. Anwer, M. Motiwala, M. Tariq, and N. Feamster. Switchblade: a Platform for Rapid Deployment of Network Protocols on Programmable Hardware. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 183–194. ACM, 2010.

[8] A. Arabo, I. Brown, and F. El-Moussa. Privacy in the Age of Mobility and Smart Devices in Smart Homes. In *Fourth IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT)*, pages 819–826. IEEE, 2012.

[9] P. Arjunan, N. Batra, H. Choi, A. Singh, P. Singh, and M. B. Srivastava. SensorAct: a Privacy and Security Aware Federated Middleware for

Building Management. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys)*, pages 80–87. ACM, 2012.

[10] J. R. Ballard, I. Rae, and A. Akella. Extensible and Scalable Network Monitoring using OpenSAFE. 2010.

[11] M. Bansal, J. Mehlman, S. Katti, and P. Levis. OpenRadio: A Programmable Wireless Dataplane. In *Proceedings of the First Workshop on Hot topics in Software Defined Networks*, pages 109–114. ACM, 2012.

[12] K. L. Calvert, W. K. Edwards, N. Feamster, R. E. Grinter, Y. Deng, and X. Zhou. Instrumenting Home Networks. *ACM SIGCOMM Computer Communication Review*, 41(1):84–89, 2011.

[13] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. A NICE way to Test OpenFlow Applications. 2012.

[14] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking Control of the Enterprise. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 1–12. ACM, 2007.

[15] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker. SANE: A Protection Architecture for Enterprise Networks. In *Proceedings of the 15th USENIX Security Symposium*, volume 15, 2006.

[16] Georgia Tech Information Security Center and the Georgia Tech Research Institute. Emerging CyberThreats Report. Technical report, 2013. `http://www.gtsecuritysummit.com/report.html`.

[17] Cisco Systems, Inc. *onePK Developer Center*, Retrieved 14 June, 2013. `http://developer.cisco.com/web/onepk/technical-overview`.

[18] R. J. Colville and G. Spafford. *Configuration Management for Virtual and Cloud Infrastructures.* Gartner Inc., 27 Oct, 2010. `http://www.gartner.com/id=1458131`.

[19] Paola Cornacchiola. *IBM Distributed Overlay Virtual Ethernet (DOVE) Networking.* virtualization.info, 14 Sept, 2012. `http://virtualization.info/en/news/2012/09/ibm-distributed-overlay-virtual-ethernet-dove-networking.html`.

[20] Symantec Corporation. Internet Security Threat Report: 2012 Trends. Technical Report 18, April 2013.

[21] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo. Software Defined Wireless Networks: Unbridling SDNs. In *European Workshop on Software Defined Networking (EWSDN)*, pages 1–6. IEEE, 2012.

[22] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler. BOSS: Building Operating System Services. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, 2013.

[23] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and V. Bahl. An Operating System for the Home. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*. USENIX Association, 2012.

[24] B. Fabian, F. Goertz, S. Kunz, S. Müller, and M. Nitzsche. Privately Waiting-A Usability Analysis of the Tor Anonymity Network. In *Sustainable e-Business Management*, pages 63–75. Springer, 2010.

[25] N. Feamster. Outsourcing Home Network Security. In *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, pages 37–42, 2010.

[26] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. R. Karger. Infranet: Circumventing web censorship and surveillance. In *USENIX Security Symposium*, pages 247–262, 2002.

[27] A. Feldmann, M. Kind, O. Maennel, G. Schaffrath, and C. Werle. *Network Virtualization - An Enabler for Overcoming Ossification.* European Community in Information Technology (ERCIM) News, Retrieved 14 June, 2013. `http://ercim-news.ercim.eu/en77/special/network-virtualization-an-enabler-for-overcoming-ossification`.

[28] J. Finkle. *'Irrational' hackers are growing U.S. security fear.* Reuters, 22 May, 2013. `http://www.reuters.com/article/2013/05/22/cybersecurity-usa-infrastructure-idUSL2N0DY1LA20130522`.

[29] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, and C. Schlesinger. Languages for Software-defined Networks. *IEEE Communications Magazine*, 51(2):128–134, 2013.

[30] A. Gember, C. Dragga, and A. Akella. Ecos: Leveraging software-defined networks to support mobile application offloading. In *Proceedings of the eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 199–210. ACM, 2012.

[31] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella. Toward Software-defined Middlebox Networking. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 7–12. ACM, 2012.

[32] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In *Proceedings of the 17th conference on Security Symposium*, pages 139–154, 2008.

[33] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an Operating System for Networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[34] A. Guha, M. Reitblatt, and N. Foster. Machine-verified Network Controllers. In *Proceedings of the 34th ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*, pages 483–494. ACM, 2013.

[35] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. Chang Shantz. Sizzle: A Standards-based End-to-end Security Architecture for the Embedded Internet. *Pervasive and Mobile Computing*, 1(4):425–445, 2005.

[36] S. Gutz, A. Story, C. Schlesinger, and N. Foster. Splendid Isolation: A Slice Abstraction for Software-defined Networks. In *Proceedings of the First Workshop on Hot topics in Software Defined Networks (HotSDN)*, pages 79–84. ACM, 2012.

[37] Hewlett-Packard. *Fact Sheet: Solutions for HP Virtual Application Networks*, Feb, 2013. `http://h17007.www1.hp.com/docs/van/vanfactsheet_4AA4-0792ENW.pdf`.

[38] C. Hoff. *The Killer App For OpenFlow and SDN? Security.* Rational Survivability Blog, 27 Oct, 2011. `http://www.rationalsurvivability.com/blog/2011/10/the-killer-app-for-openflow-and-sdn-security/`.

[39] InfoSecurity Magazine. *For Small and Medium-sized Firms, Ignorance is Not Bliss*, 17 Nov, 2011. `http://www.infosecurity-magazine.com/view/22084/for-small-and-mediumsized-firms-ignorance-is-not-bliss/`.

[40] SRI International. *BotHunter: a Network-based Botnet Diagnosis System.* `http://www.bothunter.net/`.

[41] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pages 127–132. ACM, 2012.

[42] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proceedings of the ACM SIGCOMM conference*, pages 3–14. ACM, 2013.

[43] P. Kavilanz. *Cybercrime's Easiest Prey: Small Businesses.* CNN Money, 23 April, 2013. `http://money.cnn.com/2013/04/22/smallbusiness/small-business-cybercrime/index.html`.

[44] S. M. Kerner. *SDN Market Forecast at $3.7 Billion by 2016.* Enterprise Networking Planet, 20 Dec, 2012. `http://goo.gl/pwCStU`.

[45] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 61–72. ACM, 2002.

[46] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey. VeriFlow: Verifying Network-wide Invariants in Real Time. In *Proceedings of the First Workshop on Hot topics in Software Defined Networks*, pages 49–54. ACM, 2012.

[47] Kindsight Security Labs, Alcatel-Lucent. *Kindsight Security Labs MaLware Report*, Quarter 2, 2013. `http://resources.alcatel-lucent.com/?cid=168508`.

[48] K. Lakshminarayanan, I. Stoica, and S. Shenker. Routing as a Service. Technical report, University of California, Berkeley, 2004.

[49] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor instead of IP. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets)*, 2011.

[50] S. Lui. *Case study: Ballarat Grammar uses SDN to Fight Malware*, 3 June, 2013. `http://www.zdnet.com/au/case-study-ballarat-grammar-uses-sdn-to-fight-malware-7000015942/`.

[51] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman. CrossRoads: Seamless VM Mobility Across Data Centers through Software Defined Networking. In *Network Operations and Management Symposium (NOMS)*, pages 88–96. IEEE, 2012.

[52] C. D. Marsan. *IAB Panel Debates Management Benefits, Security Challenges of Software-defined Networking.* Internet Society, Oct. 2012. `http://www.internetsociety.org/articles/iab-panel-debates-management-benefits-security-challenges-software-defined-networking`.

[53] S. McGillicuddy. *Midokura Network Virtualization: Layer 2-7 Services, OpenStack.* SearchSDN, TechTarget, 17 Oct., 2012. `http://searchsdn.techtarget.com/news/2240166952/Midokura-network-virtualization-Layer-2-7-services-OpenStack`.

[54] Shamus McGillicuddy. *Microsoft uses OpenFlow SDN for Network Monitoring and Analysis.* TechTarget, 17 April, 2013. `http://searchsdn.techtarget.com/news/2240181908/Microsoft-uses-OpenFlow-SDN-for-network-monitoring-and-analysis`.

[55] Shamus McGillicuddy. *Radware Adds Open Source DDoS Protection to OpenDaylight Project.* TechTarget, 6 Aug, 2013. `http://searchsdn.techtarget.com/news/2240203180/Radware-adds-open-source-DDoS-protection-to-OpenDaylight-Project`.

[56] S. McGillicudy. *PLUMgrid Automates Virtual Network Provisioning Without the Controller.* SearchSDN, 12 July, 2013. `http://searchsdn.techtarget.com/news/2240187981/PLUMgrid-automates-virtual-network-provisioning-without-the-controller`.

[57] 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 (CCR)*, 38(2):69–74, 2008.

[58] S. A. Mehdi, J. Khalid, and S. A. Khayam. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 6961, pages 161–180, 2011.

[59] M. Mendonca, K. Obraczka, and T. Turletti. The Case for Software-defined Networking in Heterogeneous Networked Environments. In *Proceedings of the ACM CoNEXT Student Workshop*, pages 59–60. ACM, 2012.

[60] M. Mendonca, S. Seetharaman, and K. Obraczka. A Flexible In-network IP Anonymization Service. In *IEEE International Conference on Communications (ICC)*, pages 6651–6656. IEEE, 2012.

[61] R. Meyran. *DefenseFlow: The First Ever SDN Application that Programs Networks for DoS/DDoS Security*, Retrieved 14 June, 2013. `http://blog.radware.com/security/2013/04/defenseflow-dosddos-security/`.

[62] C. Monsanto, N. Foster, R. Harrison, and D. Walker. A Compiler and Run-time System for Network Programming Languages. In *ACM SIGPLAN Notices*, volume 47, pages 217–230. ACM, 2012.

[63] R. Murty, J. Padhye, A. Wolman, and M. Welsh. Dyson: An Architecture for Extensible Wireless LANs. In *Proceedings of the 2010 USENIX Annual Technical Conference*. USENIX Association, 2010.

[64] R. Narayanan, G. Lin, A. A. Syed, S. Shafiq, and F. Gilani. A Framework to Rapidly Test SDN Use-Cases and Accelerate Middlebox Applications. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.

[65] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[66] Nicira. *Nicira Network Virtualization Platform (NVP)*, Retrieved July 14, 2013. `http://nicira.com/en/network-virtualization-platform`.

[67] M. Palmer. *Software-Defined Networking Market Expected to Reach $35 Billion by 2018*. SDNCentral, 24 April, 2013. `http://www.sdncentral.com/sdn-blog/sdn-market-sizing/2013/04/`.

[68] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A Security Enforcement Kernel for OpenFlow Networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN)*, pages 121–126. ACM, 2012.

[69] M. Reitblatt, M. Canini, A. Guha, and N. Foster. FatTire: Declarative Fault Tolerance for Software-Defined Networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking (HotSDN)*, pages 109–114. ACM, 2013.

[70] News Release. *Small Firms Lose up to 800 Million to Cyber Crime*. Federation of Small Businesses, 21 May, 2013. `http://www.fsb.org.uk/News.aspx?loc=pressroom&rec=8083`.

[71] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote Authentication Dial In User service (RADIUS), 2000.

[72] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi. The Middlebox Manifesto: Enabling Innovation in Middlebox Deployment. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets)*, page 21. ACM, 2011.

[73] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. Enabling Fast Failure Recovery in OpenFlow Networks. In *8th International Workshop on the Design of Reliable Communication Networks (DRCN)*, pages 164–171. IEEE, 2011.

[74] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson. FRESCO: Modular Composable Security Services for Software-Defined Networks. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2013.

[75] H. Solomon. *Shodan, Called 'the Scariest Search Engine on the Internet', Finds Traffic Lights, Power Plants.* News Ltd., 10 April, 2013. `http://www.news.com.au/technology/shodan-called-the-scariest-search-engine-on-the-internet-finds-traffic-lights-power-plants/story-e6frfro0-1226616893647`.

[76] H. Solomon. *Networks Face 'Crisis of Trust': Websense.* it World Canada, 13 Feb, 2013. `http://www.itworldcanada.com/news/networks-face-crisis-of-trust-websense/146736`.

[77] S. Son, S. Shin, V. Yegneswaran, P. Porras, and C. Gu. Model Checking Invariant Security Properties in OpenFlow. 2013.

[78] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. Towards Programmable Enterprise WLANS with Odin. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN)*, pages 115–120. ACM, 2012.

[79] R. Tagnipes. High Availability with Dynamic Load Balancers. 4 Feb, 2013. `http://blog.gogrid.com/2013/02/04/high-availability-with-dynamic-load-balancers/`.

[80] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On Controller Performance in Software-defined Networks. In *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2012.

[81] J. Whiteaker, F. Schneider, R. Teixeira, C. Diot, A. Soule, F. Picconi, and M. May. Expanding Home Services with Advanced Gateways. In *ACM SIGCOMM Computer Communication Review*, volume 42, pages 37–43. ACM, 2012.

[82] A. Williams. Ciscos Lew Tucker on the Internet of Everything and the Tie to an App-Centric World. 3 April, 2013. `http://techcrunch.com/2013/04/03/ciscos-lew-tucker-on-the-internet-of-everything-and-the-tie-to-an-app-centric-world/`.

[83] K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. The Stanford Openroads Deployment. In *Proceedings of the 4th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 59–66. ACM, 2009.

[84] M. Yu, L. Jose, and R. Miao. Software Defined Traffic Measurement with OpenSketch. In *Proceedings 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, volume 13, 2013.