A Review of Solutions for SDN-Exclusive Security Issues

Jakob Spooner
Department of Computing and Mathematics
University of Derby
Derby, England

Dr Shao Ying Zhu
Department of Computing and Mathematics
University of Derby
Derby, England

***Abstract*—Software Defined Networking is a paradigm still in its emergent stages in the realm of production-scale networks. Centralisation of network control introduces a new level of flexibility for network administrators and programmers. Security is a huge factor contributing to consumer resistance to implementation of SDN architecture. Without addressing the issues inherent from SDNs centralised nature, the benefits in performance and network configurative flexibility cannot be harnessed. This paper explores key threats posed to SDN environments and comparatively analyses some of the mechanisms proposed as mitigations against these threats – it also provides some insight into the future works which would enable a securer SDN architecture.**

***Keywords*—SDN; software; security; OpenFlow; networking; network security; NFV**

I. Introduction

Software Defined Networking is a paradigm that emerged in around 1995 with the introduction of Active Networking – programmable functions integrated within network architecture, enabling programmers to innovate the way in which they function [1]. Whilst the roots of SDN lay in technologies first introduced over 20 years ago, the concept is still extremely relevant to this day and is considered by many to be the new face of networking [2]. Historically, packetised data-networks have consisted of hardware-based networking devices operating at Layers 2 and 3 of the OSI model. Software is then implemented on top of these layers, to provide other pieces of vital network functionality, i.e. transport control for the network, e-mail applications, file transfer etc.

In traditional networks, network hardware such as routers and switches can be logically divided into two individual planes; the data plane and the control plane. The data plane is concerned with the forwarding of data-packets, whilst the control plane makes packet-forwarding decisions based on the routing protocols configured on the device. The tightly bundled nature of these two planes introduces a level of rigidity – network operators cannot easily manipulate forwarding decisions on a per flow basis. SDN aims to challenge this by separating the control and data planes. This segregation allows network programmers to develop their own controllers, pieces of software with a global view of the network [2]. This allows for a level of control that was not possible without a great deal of work in traditional network architectures due to the tightly bundled nature of data-plane and control-plane. In SDN, rules, known as flows, based on a set of conditions (e.g. all HTTP packets over a particular size) are created centrally by network admins, installed on the controller and then pushed out to network devices in the data plane. Devices store the flows in their local cache, and in the event that they receive a packet, they check the currently stored flows for one matching the received packet. These flows govern the way in which packets should traverse the network [3], leading to a network which is easier to manage due to a centralisation of control.

SDN gives network administrators the ability to collect traffic statistics from the network devices and pass these onto the control plane for processing. This allows for in depth security-analysis without any negative effects on the performance of the data-plane [4]. SDN makes it possible to configure security policies centrally at the controller and push them out network wide. This is in stark contrast to the painstaking process of individually configuring access control lists and security policies on every router or switch in the network [5]. SDN allows easy integration of third-party software into the environment via the SDN framework, meaning that plugin-like applications can be deployed to aid certain security & non-security related tasks [4]. As SDN controllers hold a global view of the network, they introduce the possibility of network-wide intrusion detection systems, which utilize the traffic statistics they receive from the network devices. As devices are required to communicate back to the controller at regular intervals, it ensures that compromised devices are found quickly and reduces the chances of false positives, an issue that is still yet to be solved in the context of traditional networks [3].

Whilst some of the benefits of an SDN-based infrastructure are clear, there are also some apparent shortcomings, which need to be addressed before implementation of the paradigm can become widespread. Programmers are able to leverage the centralised control in SDN architectures to build reactive, self-healing mechanisms to mitigate against traditional network attacks [6]. However, the fact that SDN changes the way that networks operate entirely is likely to bring about new attack methods that can be used to exploit the individual components of an SDN architecture, and the ways in which they interact (i.e. devices-to-controller, controller-to-controller and controller-to-application). For example, an attacker successfully compromising the controller of a network is particularly lethal, as this single-point of failure can render the entire network inaccessible [4].

The introduction of a centralised controller completely changes a networks architecture. This is what makes SDN so unique in comparison to traditional networks. With the centralised point of control, all other layers need to maintain an interface over which they can exchange important information. Commonly, the interface utilised by the data-plane and control-plane to communicate with one-another is known as the Control-Data-Plane interface, or the Southbound Interface. SDN Applications also reside in a conceptual application layer, and communicate with the controller through the 'Northbound Interface' or Control-Application-Plane interface. Applications residing on this layer have the ability to solicit directly with the controller and obtain useful information about the networks logical/physical state. This is advantageous to programmers writing SDN applications, as their programs can access large quantities of meaningful, real-time data. With this, however, comes great risk – adversaries may be able to program their applications to utilise this useful information to form attacks, and compromise the availability, integrity and confidentiality of data travelling within the network.

This change in network architecture brings around juxtaposition. On one hand, the increased flexibility and ability to innovate with network applications and network control, gives programmers the ability to better protect against traditional network attacks i.e. TCP-based DoS attacks, eavesdropping, man-in-the-middle attacks. On the other hand, the links between control-plane, data-plane and application-plane bring about new attack platforms for adversaries attempting to illegitimately use network services. Much research has been carried out over the years on traditional security attacks, however this paper focuses on attacks which are exclusive to SDN due to changes in the architecture. The solutions covered below attempt to mitigate attacks targeting these SDN-exclusive attack platforms, and ensure that adversaries cannot utilise the change in architecture to their own advantage.

The remainder of this paper is structured as follows: section II explores some of the security threats aimed at SDN environments; section III discusses some of the currently proposed mitigations and provides a comparative analysis of them; section IV discusses the current gaps in security and how these can be filled going forward, and section V concludes the paper, with an insight into future direction

II. Threats to Security in SDN Environments

Networks running under the SDN paradigm still have the same security requirements as traditional network settings, as it is likely that they will be carrying at times, private and confidential information [7]. SDN completely changes the architecture and the inter-communicative aspects of the components in the network - from this arises a completely new platform for attackers looking to perform security-breaching attacks. This leads to a need for similar levels of security as traditional networks, but to defend against threats of a different nature [8]. This section of the paper examines some of these key threats and aims to justify their importance.

A. DDoS/DoS Attacks (Flow-decision Requests)

1) DDoS (Flow-decision Requests): Numerous types of conventional DDoS attacks can be carried out in an SDN environment, but it is a variation utilising forged flow entries which can be harnessed by an adversary in order to target a controller and compromise its availability. By flooding the controller with requests for a flow-decision, the controllers compute resources could become overwhelmed, and the controller would be rendered unable to deal with any legitimate requests it receives [2]. By targeting the centralised point of control (i.e. the controller) it renders the entire network largely unusable. Whilst data-paths currently in the network may be able to function temporarily with a downed controller, once the hard timeout of rules in their table has expired they will be required to solicit with the controller again, which will be unable to deal with requests. If an attacker(s) is able to be persistent with their flooding, this will eventually cause the unavailability of all network functionality.

2) DoS Attacks (Switch flow-table entry flooding): At the data plane level, falsely created flow-entries can be flooded to other devices in order to consume the space in their flow entry tables. This leaves the forwarding devices unable to add any legitimate flow entries to their tables [3]. This results in devices being unable to incorporate subsequent flow-updates, leaving the network in a broken, disparate state. One of the key issues with the data-plane devices within Software Defined architectures is that of the switches inabilities to differentiate between legitimate flow requests and illegitimate ones. This flaw allows for attackers to perform successful DoS attacks at the Data plane level by filling the switches flow-buffer with false requests [4].

Whilst it would be possible for an adversary to target an individual data-path and attempt to halt its availability, it is far more likely that the controller would be targeted, effectively creating and spreading a system-wide lapse in availability. The prospect of this can be potentially devastating, particularly in production settings where services seeing high usage will be unusable to clients and employees. Furthermore, with availability for all clients removed, an adversary can plan and carry out further attacks which may aim to compromise the integrity and confidentiality of sensitive data on the network. For these reasons the above DDoS/DoS attacks have been mentioned in this paper and are considered amongst the most important attack types.

B. Hijacked/Rogue Controller

The controller can be thought of as the centralized ‘brain’ of an SDN. It controls the whole network from one point, making it arguably the most vital component of SDN architecture. An attacker that manages to compromise the controller essentially has control over the whole network [9]. Ability to control the actions of the controller would allow the attacker to manipulate flow entries in any way that he/she choose, e.g. stopping certain packet types reaching their destination, re-directing packets to malicious nodes in the infrastructure. In conjunction with this, the attacker could aim to compromise a particular forwarding-device in the network and enable it to operate as a ‘man-in-the-middle’ or black-hole/grey-hole node. This would allow the attacker to potentially drop, alter or inspect the contents of any packet it receives [10]. Another possibility is that an attacker successfully registers a 'rogue' controller in the control plane of the network. With this rogue controller in place the adversary may be able to influence/halt the availability of other controllers, change rules installed in data-paths caches and effectively halt/manipulate the workings of applications in the applications layer.

Any attack that targets the controller/controllers in SDN architecture can have potentially devastating effects. Whilst the centralised nature and ability to collate information at one point can be massively advantageous to network administrators/programmers, in the wrong hands it could be utilised to spearhead attacks on the integrity of control messages/sensitive application information, the availability of important services to a systems users, and the confidentiality of sensitive user information utilised by applications in the application layer. Any approach attempting to successfully halt hijacked/rogue controllers should focus on ensuring the authenticity of the controller, before allowing it to make any changes to the network.

C. *Malicious Applications*

Due to the allowance of the SDN framework for integration of third-party applications, the issue of malicious applications arises. Applications exhibiting malicious behavior within an SDN environment can have catastrophic consequences, similar to that of a compromised controller [2]. Authentication and authorization of an application to operate within an SDN environment is difficult to enforce. Applications relying on deep packet-inspection techniques to operate can pose potential risks to the network – they may be able to indirectly control the entire network through the information they have collected during packet-inspection [11].

The increased amount of data, and the way in which it is centrally located is what gives malicious applications the ability to threaten the integrity and confidentiality of user/network information that they have access to. Securing the northbound interface is a difficult task, as each application utilising it may require access to a unique subset of information from the controller. In order to successfully monitor this, some kind of strict, information-access policy need to be enforced. This ensures that an application declares which information it will need and is only able to access these. This could ensure that applications are not covertly stealing or using information from other applications. Authenticity must also be ensured, before an application is able to communicate with the controller.

D. *Control-Data Plane Link Attacks*

Another key area in SDNs which presents opportunities for attackers would be the link between the control plane and the data plane. The OpenFlow specification defines use of TLS (Transport Layer Security) as optional [12], making this a weak-point and clearly susceptible to various attacks, i.e. man-in-the-middle attacks, black-hole attacks.

1) Man-in-the-middle Attack: A man-in-the-middle type attack takes place when a malicious node establishes itself between the controller and the data-paths residing on the data plane. Instead of directly forwarding the messages straight to the controller (or vice-versa), the 'man-in-the-middle' node is able to manipulate/ or inspect the contents of packets [13].

2) Black-hole Attack: A black-hole type attack could also be performed, in which a node establishes itself in between a targeted device and the controller, and simply drops any packets it receives without forwarding them to the controller. This results in a breakdown of network communications and renders the services unavailable to legitimate users [3].

If an attacker does manage to establish itself as an intermediary between the control plane and data plane, it can potentially be devastating to the entire network. The man-in-the-middle type attack is a direct attack on the integrity of control messages between network devices in the data plane and the controller. An adversary can change control messages and shape the way the network is formed to a way advantageous to them. On the other hand, the black-hole type attack is a direct attack on the availability of the networks services. If all messages between network devices and the controller are not being forwarded by the malicious node, it will inevitably result in a breakdown in communication, with devices in the data plane unable to solicit the controller when necessary. This link between control and data plane is clearly a weak-point, and acts as a ripe attack platform for adversaries. It is therefore extremely important that it is secure before SDNs see widespread usage in production settings.

E. *Eavesdropping Attacks*

Adversaries attempting to gain illegitimate access to SDN networks or halt service availability may find it advantageous to eavesdrop (the act of illegitimately capturing and inspecting the packets flowing over a connection) on certain connections in the network [14]. This may allow them to gather meaningful information which can then be used to carry out more intrusive attacks. Eavesdropping attacks have long been carried out in traditional network settings – wireless architectures are particularly weak due to their over-the-air transmissions. However, in the context of SDN, eavesdropping can be carried out to inspect the packets traversing the link between control-data plane, and also exclusively at the data plane. At the data plane, [15] discusses a ease-of-use 'listening' mode integrated into OpenFlow switches can be utilized by a malicious adversary (that has been able to compromise the switch) in order to inspect the packets transmitted by surrounding switches, allowing attackers to learn important control information. In a sense, eavesdropping carried out at the control and data plane is more of a passive attack and does not directly affect the availability, confidentiality or integrity of data. It does, however, empower attackers to carry out further attacks which compromise these security aspects.

In the context of eavesdropping carried out at the Application-Control Plane link, however, the confidentiality of sensitive information can be directly compromised. A malicious adversary can learn information pertaining to a particular user if they manage to eavesdrop on a connection transmitting sensitive application data. This makes eaves-dropping a particularly serious attack – confidentiality must be ensured before critical applications carrying sensitive information can be deployed in an SDN-environment.

F. Side by Side Comparison of Attacks

The following table summarises the above investigation and allows us to see the layers of abstraction that each attack affects in the SDN-architecture, and the specific security aspects that they could potentially compromise.

TABLE I. COMPARISON OF SDN ATTACK TYPES

Attack	**Targeted SDN Layer**	**Affected Security Aspect**		
		Availability	***Confidentiality***	***Integrity***
Distributed Denial of Service	Control, Data	x		
Denial of Service	Control, Data	x		
Hijacked/ Rogue Controller	Control, Data, App	x	x	x
Malicious Applications	App		x	x
Man-in-the-middle	Control, Data, Control-data link		x	x
Black-hole	Control, Data, Control-data link	x	x	
Eavesdropping	Control, Data, App		x	

Each attack investigated in this section has been, or still is, fairly common in the realm of traditional networks. The interesting part is, that with the introduction of new attack-platforms inherent from SDNs architectural changes, comes variations of the attack which are then exclusive to SDNs. Whilst the traditional variations of each of these attacks can be dealt with in a more appropriate, effective manner due to the centralised nature of SDN, it is these SDN-exclusive variations which pose the largest challenge and that need to be given attention when moving forward with SDN security. The following section aims to look at currently proposed solutions to the above attacks and evaluates their effectiveness.

III. MITIGATIONS AGAINST SECURITY CHALLENGES IN SDN

The previous section defined some of the key threats to both the control and data planes in the context of SDN environments. The separation of these planes leads to highly configurable networks; however it also introduces the possibility of a number of security threats. This section explores some of the mitigations proposed for these individual threats, and then introduces some network-wide solutions which aim to secure both the control and data plane [2].

A. DDoS/DoS Attack Mitigations

DoS Attacks on SDN networks can be carried out at both the control and data plane levels. Below are a number of solutions; some specifically aim to defend the control plane, some the data plane, and others provide protection to both of these planes.

Seungwon Shin et al. [16] introduce a solution for TCP based control plane DoS attacks – AVANT-GUARD. This solution consists of two components; a Connection Migration mechanism used in establishing useful TCP sessions from failed ones, and Actuating Triggers which enable data plane devices to activate flow rules under certain pre-defined conditions. Connection Migration proxies the TCP handshake that takes place when nodes initiate a TCP connection, and ensures that the handshake is successfully completed and the session established before allowing any flow entries pertaining to this session to be forwarded to the controller. This reduces the possibility of TCP-SYN packet flooding attacks on the controller as the handshake will not have been completed for these sessions. The Actuating Triggers mechanism introduced in [16] also reduces the computational load incurred by the controller, by allowing devices to activate certain flow rules in their tables under predefined conditions. This reduces the number of transmitted flow-requests. Evaluative tests prove that in the presence of a TCP-SYN based DDoS attack, the response time of the controller to legitimate flow requests increases by a negligible amount in the order of milliseconds, along with the percentage of overhead incurred [16]. Whilst the performance of AVANT-GUARD is desirable, the approach is generally limited due to the fact that it targets one particular variation of the DDoS attack (TCP-SYN based attacks). An option would be to deploy this solution alongside other DDoS prevention mechanisms targeting other attack variations, however this configuration could become cumbersome and resource intensive. Instead, a solution defending the control plane from a wide variety of DDoS attack types would be more desirable.

Paulo Fonseca et al. [17] present a novel mechanism to prevent control plane based DoS attacks - CPRecovery. This replication based component allows for the handing-over of control from one controller, failing due to the presence of a saturation attack, to a secondary controller. To achieve this, switches check for the presence of a properly operating controller by sending an inactivity probe. If this probe determines inactivity in the controller, a connection is made to a secondary controller which assumes the role of the failed one. To ensure this process is seamless, the initial primary controller sends state update messages to the secondary controllers in the network [17]. This multi-layer approach does provide resilience, and unlike AVANT-GUARD [16], it provides the control plane with protection from a wide variety of attacks. It is, however, noted in [17] that once the secondary controller assumes a primary role, it is then susceptible to DoS attacks itself. One further downside to CPRecovery is its performance. Due to the overhead incurred from the instantiation and connection of recovery controllers, CPRecovery can be quite slow [17], with response times far higher than that of AVANT-GUARD. It would appear that this is a trade-off for the more thorough protection offered in [17].

Presented in [18] is FlowRanger, a proposal described as a request prioritizing algorithm for control plane-based DoS attacks. At a fundamental level, FlowRanger implements a priority-based scheduling system, with its key metric being that of trust-values held by each node in a network. Controllers implementing FlowRanger are able to evaluate the trust values of each node they are receiving requests from and buffer them in separate priority queues [18]. Other proposals ([16], [17]) implement a rate-limiter for the amount of requests which can be sent to the controller, however this results in the dropping of some legitimate requests. This is clearly an issue; in production environments it is not acceptable for legitimate flow-requests to be dropped. FlowRanger challenges this with its priority queueing mechanism; suspected attacking requests will still be served, albeit at a lower priority than others. This works well, for example, if a legitimate switch is having issues and is having to retransmit flow-requests to the controller. It will be initially buffered in a lower priority queue, but eventually the request will be served. This means that once the switch has established itself and replenished its flow-cache with correct rules, its trust value will increase and its requests will return to higher priority queues. FlowRanger differs from previous implementation in that its priority queues are implemented at the controller, instead of in a distributed manner. This provides a further layer of protection (as long as the controller is not compromised). Simulation results also show that the centralised nature of this mechanism make it a higher performer than previously proposed solutions [18]. Reference [19] introduces a method utilizing user-behaviour analysis. Whilst this sounds effective in premise, the adoption of strong assumptions regarding the number of flow-requests generated by users leaves a lot to be desired. The source IPs of requests is tracked by the controller and if the profile of packets received from that IP fall into a certain category, the IP is marked as malicious. The controller then subsequently drops requests from that IP. This suffers from similar issues to AVANT-GUARD [16], in which genuine requests from non-malicious users may be ignored.

References [16], [17], [18] and [19] all focus on control plane-based DoS/DDoS attacks. Whilst it is arguably more vital that the control plane is protected, data-plane based attacks do exist and require ample protection. In [20] a data-plane oriented DoS mitigation known as Virtual source Address Validation Edge mechanism (VAVE) is proposed. VAVE is pre-emptive as it attempts to detect the presence of a node using IP-spoofing techniques to mask its real identity – this type of behaviour often leads to DoS attacks. By checking incoming packets against entries in the flow-table, the VAVE interface in the network determines if the packet is a recognised type – if not, the validity of the packet is checked against a list of pre-defined rules [20]. This could potentially cause issues, as if the pre-defined list of rules is not thorough, certain legitimate packets could be dropped, causing availability issues for genuine clients. As this mitigation only defends against data plane attacks, it would leave the controller susceptible to DoS attacks. To provide thorough protection against all types of DoS, parallel implementation of VAVE alongside AVANT-GUARD or CPRecovery would be more appropriate. This parallel implementation does however come with the downside of high computational cost and a chance of conflicting configurations.

One major issue with the currently proposed solutions for DoS/DDoS attacks is that they wait for the adversary to strike, and then attempt to deal with the aftermath. In the case of [16], [17] and [21], it is possible for the network to be rendered temporarily unavailable before any attempt to mitigate is made. FlowRanger [18] challenges this to a degree, and in general appears to be the strongest solution from the selection reviewed above. One solution which does put emphasis on early DDoS detection is presented in [22]. The scheme utilises a measure of entropy variation in the attack packets destination IP field. The method claims to stop the attack within the first 500 transmitted attack packets. Whilst this is early, it is likely that a reasonable amount of damage will have been done during this period of time. It seems clear that defending against attacks in the control plane are two entirely separating things. Networks would benefit greatly from a solution which simultaneously protects both of these planes, whilst still achieving a reasonable level of performance.

B. *Hijacked/Rogue Controller Mitigations*

Unauthorised access to the controller in an SDN environment has been identified as one of the most potent threats. Various approaches towards protecting an SDN from unauthorised access at the control plane level have been proposed. The authors in [9] suggest multiple-controller architecture coupled with a ‘Byzantine Fault-Tolerance’ mechanism, in which a number of controllers dictate to the switches in the environment. Each switch in the network is connected to a set of controllers, and when one controller fails due to a successful attack, another controller takes over and connection with the failing one is severed. This particular proposal utilises a high level of resources; however an algorithm is introduced to reduce resource consumption by determining the optimal amount of controllers connected to each switch based on latency requirements and the tolerance of the switch to faulty controllers. Simulation results presented in [9] suggest that the method achieves reasonable levels of performance when the size of the data plane is small; however, as the number of switches grows the amount of controllers needed also increases, resulting in lower performance in larger-scale environments. It could be said that the proposal in [9] suffers from the same flaws as CPRecovery, in that even though a replacement controller gains control of the network, once it has taken control it is then subject to the same attacks as the previous controller. It would appear that the most effective solutions provide defense against the occurrence of an attack, and not just provide fault-tolerance and resilience should the attack occur.

In contrast to the above proposal, the authors of [23] introduce a mechanism aiming to secure SDNs by forcing them to operate in a distributed manner. An earlier work, [24], presented by the same authors defines the workings of this hybrid, distributed SDN system, in which the controller continues to centrally define flow rules, but puts algorithms in place to enable switches to spread flows to other devices in the network. In [23], a number of mechanisms are proposed which work cooperatively to provide security in this distributed environment. The use of a Trust Manager System is introduced. This is an actively maintained list containing ID’s for all of the devices currently operating in the network. This is then coupled with encrypted transmissions of all flow entries occurring between controller- data plane device and data plane device– data plane device. An authentication mechanism is put in place to allow each data plane device in the network to check the authenticity and validity of the node that a flow originated from. This approach is in stark contrast to [9] which relies simply on a fault-tolerance mechanism. As mentioned in section II.C.2 of this paper, [23] enforces the use of TLS in all of its controller-to-equipment transmissions, making these communications inherently secure. One inherent issue with the distributed nature of [23] would be the difficult process of configuring and debugging issues with it. Since the overall proposal consists of a number of linked mechanisms, a fault in one place would bring down the entire system, or at least leave it partially operating and open to attacks. The process of accurately pinpointing the point of failure could be a difficult process. Since the nature of SDN forces networks to operate in a centralised manner, it would seem logical to develop centralised security mechanisms to complement it.

No simulation results are presented in [23] leading to the assumption that the method may not have been subject to adequate testing as of yet - this would make [9] seem more reliable – however, the lower computational overhead incurred in [23] would make it more desirable in the network setting. The distributed architecture presented also appears to be more efficient, and its generic nature allows the use of any encryption method to be used as part of its implementation, making it a more flexible option [23]. One similarity between these two solutions is the fact that they both provide protection only to communications existing on the control plane and the control-data plane link. Presented in [25] is a method which provides protection for both the Control Plane and the Data Plane. AuthFlow prevents the access of unauthorised hosts to the network, and ensures that they are properly authenticated through the use of a RADIUS server. Extensible Authentication Protocol (EAP) is used to encapsulate any messages that hosts send to the RADIUS server, requesting authentication. An intercepting authenticator relays these messages to the controller, which then adds the host sending the packets to its list of authorised hosts. A particular strength of this proposal is that packets containing flow entry updates for other devices are not transmitted until both devices are successfully authenticated [25]. Whilst this approach provides strong protection and authorisation for legitimate hosts on the data plane, its weakness lies in its underlying technologies. EAP is generally considered as weak in the realm of authentication [26], and whilst it does allow the configurators to select their own authentication and encryption methods, any hacker with knowledge in traditional attacks should be able to force an entry into the network as a rogue data-path.

Compared to the Byzantine Fault Tolerance mechanism described in [9] and the distributed security model in [23], AuthFlow appears to offer a more thorough solution, providing authentication and authorisation at both the control and data planes of an SDN environment. Test results presented in the AuthFlow paper suggest that the method successfully prevents unauthorised access of both data-plane hosts and controllers to the network, and does so in an efficient manner - with low computational and communicational overhead incurred, due to the low amount of controller input [25]. The AuthFlow mechanism also appears to be more scalable than the other two proposals in that it does not require more controllers to be added as the data plane increases its size; new devices can simply authenticate with the RADIUS server and be added to the trusted-device list stored at the controller.

One further proposal which appears to offer strong authenticity, validity and integrity in terms of the flow rules installed in switches is PERM-GUARD [27]. Offering protection to at all layers of the SDN model (Control, Data, Application, and the links in between them), PERM-GUARD employs a scheme that manages the flow-rule production permissions of controllers and applications in an SDN infrastructure. If a controller or application wishes to push flow-rules out to data-paths on the network, they are required to authenticate themselves to a centralised authority by means of an identity based signature. Each legitimate controller or application will hold one of these signatures, and appropriate flow-production permissions will be set on authentication [27]. If one of these signatures cannot be presented, then the controller/application in question will be considered illegitimate. Whilst this does not necessarily stop attackers from attempting to hijack controllers or create rogue controllers and connect them to the network, it will deny them the ability to make malicious changes to the network structure by pushing rules out to data-paths in the network.

It would appear that PERM-GUARD is the strongest of all the solutions covered in the above section – it provides protection for not only controllers but also applications. It is a great strength of any solution when it is able to provide cross-layer protection from a common attack. In general it appears that it is not the task of stopping an already-identified hijacked/rogue controller that is difficult - it seems detecting them in the first place is the hard part. Preemptive mechanisms which prevent hijackers and rogue controllers from ever gaining access to the network would need to be developed in order to ensure that there is little to no chance of compromise.

C. Malicious Application Mitigations

Malicious applications are another dangerous threat in the context of SDNs. A compromised application or an application that has been programmed with malicious intent can allow an attacker to leverage control of the entire network [28]. Similarly, an application with buggy code can introduce vulnerabilities that attackers could exploit to gain unauthorised access to the network. The authors in [21] propose ‘FortNOX' – a mechanism which monitors the insertion of flow-rules from security applications to devices in the network. New flow-rules are first checked against all other existing flow rules on a receiving device, and in the presence of a conflict, the new flow is discarded and not added to the local cache. This mechanism allows admins to enforce a set of hard-coded rules which override any dynamically-created rules. This ensures false flows added to the network by malicious applications to direct traffic to compromised nodes will not be permitted [21]. One particular downfall of this approach is the fact that legitimate security-applications that rely on dynamic modification of flows within the network will not operate properly in the presence of admin-created hardcoded rules. The authors in [21] do not explore this scenario, so future work could relate to this area and introduce a permissions-based mechanism to allow certain dynamic flows to take precedence.

An alternative to the previous proposal, the authors in [28] present Rosemary - a small network operating system (NOS) which is ran in multiple instances on top of the control plane of an SDN. Each application running in the environment is executed within an individual instance of Rosemary, effectively isolating each application. This allows for close monitoring of every application deployed on the network in terms of the resources it uses and the packets it transmits or receives. As well as providing a platform for monitoring malicious activities of individual network applications, Rosemary provides a level of resilience in that if one application fails or crashes, others will continue to run inside their isolated instance of the NOS [28]. Rosemary's description as a 'robust, secure and high-performance NOS' would concur with the results of evaluative test. Tests show that successful attack rates are very low in the presence of Rosemary, with high levels of performance achieved in terms of the throughput achieved and latency observed. This, however, comes with the downside of a high computational overhead, especially when compared to FortNOX which incurs relatively low levels of overhead [21].

Another proposal, LegoSDN [29] is conceptually similar to Rosemary in that it provides a layer of isolation between the controller and the application layer of an SDN environment. The key difference is that all applications are bundled together in one plane, whereas Rosemary implements a single container for each individual application. Whilst LegoSDN isn't designed specifically for combatting attacks, it isolates applications that are identified as failing – a failing application being one of the key signs of the beginnings of an application-based attack [29]. In this sense, LegoSDN provides a pre-emptive mechanism – isolating potentially malicious apps from the network before they are able to cause damage. In comparison to FortNOX and Rosemary, LegoSDN does not provide a thorough enough mitigation against attacks. A parallel implementation of LegoSDN and FortNOX may be more appropriate, providing a type of first line and second line defence, respectively.

OperationCheckpoint is introduced in [10] as an SDN application control system, taking into account both the information that an SDN application reads from the underlying network, and the rules and policies it pushes out to data-paths in the network. OperationCheckpoint employs a thorough set of user-defined permissions, covering all OpenFlow related tasks. Each application that then needs to be used will be mapped to a specific subset of these permissions. Applications will then be unable to perform any actions that lie outside the set of permissions it has been mapped to (unless a change is authorised by administrators/operators) [10]. This effectively stops applications created by malicious adversaries from capturing sensitive data or executing malicious commands in the data plane, because these actions need to be declared first. By enforcing this declaration of necessary functionality, it allows network operators to build bespoke permission sets and ensure they know precisely how each application is behaving.

OperationCheckpoint seems to be the most thorough method of protecting an SDN infrastructure from malicious applications. In comparison to other solutions, it seems to perform better due to the simplistic nature of its permissions system. This is in contrast to certain solutions such as Rosemary, which in terms of its performance is lacking due to the fact that every application is ran in its own environment. This has the tendency to incur high levels of overhead. One feature of SDN is the fact that there are a set of common APIs that can be used at the northbound interface between applications and the controller. This is advantageous in the sense that it removes the possibility of cross platform vulnerabilities arising through poor configuration/coding of middleware.

D. Control Plane and Data Plane Link Attack Mitigations

The link between the control plane and data plane carries transmissions of flow-entries down to the data plane for devices to add to their local cache [28]. The specification for OpenFlow, the most widely used and supported protocol for SDN operations [2], specifies an optional implementation of Transport Layer Security (TLS) for this link, however as this is optional, many of the available controller APIs don't enforce or support this option. For this connection to be completely secure, controller specifications would need to enforce this usage of TLS – providing secure encrypted transmissions and the authentication of entities at each end of the link.

In addition to this, IDS style systems could be implemented to thwart man-in-the-middle and black-hole type attacks occurring on this particular link. The authors in [30] present a methodology in which the Bro IDS system is integrated with a Ryu-based Python controller. On reception of a packet the IDS utilises deep packet inspection techniques to inspect the packet and determine its source, destination, payload among other things. Acting as a signature based IDS, bad-traffic and malicious activities are pre-configured into the controller, and unknown packets are then checked against these signatures to identify the presence of malicious packets [30]. Evaluative tests carried out in [30] suggest that the response times of this mechanism are quite slow. This is due to the additional load incurred on the controller from integration of the IDS system.

Due to the fact that the majority of attacks that take place within this link are variations on the traditional man-in-the-middle and black-hole attacks, it may be possible to protect the Control-Data plane link via traditional means. That is not to say that this would be a thorough solution. As explored above, employing an IDS system to identify potentially malicious nodes forwarding messages across this link would ensure that falsified messages are not forwarded between data-paths and controllers, confidential data is not extracted, and that availability is not compromised by means of a black-hole attack.

E. Eavesdropping Attack Mitigations

Defending against eavesdropping in any environment can be a particularly difficult task. This is due to the passive nature of the attack; attackers can simply establish themselves as a node in the network and activate a listening mode. This enables them to view the stream of packets that traverses them. Since eavesdropping nodes will often appear to have similar behavioural characteristics to legitimate clients, it can be difficult to detect them. In an SDN environment, this difficulty is exacerbated with the introduction of new attack platforms stemming from the centralised nature. An example of this would be the control-data plane link and the control-application plane link.

A passive approach to defending against eavesdropping in the data plane and the control-data plane link is presented in [26] as Random Route Mutation. Its premise lies in randomly changing the course of packet flow, such that the destination IP stays the same. By changing this flow it aims to obfuscate the packet traces collected by adversarial clients. Such a method has been implemented in traditional network settings with moderate success. Authors in [26] claim that simulation results prove the RRM mechanism to be both efficient and effective in reducing the number of successfully eavesdropped packets. The approach has also been implemented in a NOX controller, suggesting that it is applicable to SDN environments as well. One particular downfall of this approach is it does not consider the intelligence of the adversaries it attempts to defend against. Whilst the technique may instantly see considerable reduction in the amount of eavesdropped packets, should adversaries learn of the algorithms utilized for randomizing packet flows, they may be able to reverse the affects by applying the reverse algorithm to their packet captures. Whilst this passive approach provides some mitigation, it is clear that its effect is limited and a more thorough solution would be necessary should SDN see widespread usage in production settings.

In [15], Combat-Sniff is introduced. Combat-Sniff implements both an active-detection mechanism which actively scans for eavesdropping nodes, and a pro-active defense method which aims to prevent malicious adversaries from being able to sniff packets in the first place. An attacker is able to force a switch into storing an illegitimate flow entry in its cache that forwards all packets to itself. Combat-Sniff combats this by taking random samples of the flow entries installed in each of the switches tables and checking their integrity [15]. Should a flow be identified as illegitimate, the appropriate port on the switch is shut down and packets are no longer transmitted through it. Combat-Sniff also aims to retain the confidentiality of information on packets travelling through a switch by ensuring that the switch is partially blind. This means that the switch is intelligent enough to know how to forward the packet, but is unaware of the packets contents. Whilst, in premise, this solution seems technically sound, and evaluative tests prove it to be a reasonable performer with a fair level of effectiveness, it would be important to ensure that the weight of the random flow-entry sampling is ample enough to ensure illegitimate flows are identified.

It appears that eavesdropping is generally a very difficult attack to defend against, with not many solutions existing solely to defend against it. Intrusion Detection Systems and Intrusion Protection Systems can potentially be configured to detect eavesdropping nodes, however the passive nature of the attack makes it inherently difficult to detect. It appears that the two existent solutions ([26], [15]) are both more geared towards reducing the damage of any eavesdropping attacks. A more effective solution would attempt to prevent any sort of eavesdropping taking place. This could be achieved by employing some strong encryption method, and giving only authenticated data-paths and controllers the correct key to unencrypt packets.

TABLE II. SUMMARY OF ATTACK MITIGATION

Targeted Attack	**Affected Security Aspects**	**Proposed Solution**	**SDN Layer**				
			Data	***Control-Data link***	***Control***	***Control-App link***	***Application***
DoS/DDoS	Availability	AVANT-GUARD [16]			x		
		CPRecovery [17]			x		
		FlowRanger [18]			x		
		VAVE [20]	x				
		Entropy-based detection [22]	x		x		
Hijacked/Rogue Controller	Availability, Confidentiality, Integrity	Byzantine Fault Tolerance [9]			x		
		Trust Management System [23]	x		x		
		AuthFlow [25]		x	x	x	x
		PERM-GUARD [27]	x		x		x
Malicious Applications	Confidentiality, Integrity	FortNOX [21]		x	x	x	x
		Rosemary [28]			x		x
		LegoSDN [29]			x	x	x
		OperationCheckpoint [10]			x	x	x
Control-Data Link Plane Attacks (MITM, Black-hole)	Availability, Confidentiality, Integrity	BroIDS [2]	x	x	x	x	x
Eavesdropping	Confidentiality	Random Route Mutation	x	x			
		Combat-Sniff [26]	x	x			

F. Summarising solutions

Each of the attack solutions explored above have been evaluated in terms of their strengths and weaknesses. Solutions have been comparatively analysed in order to identify the gaps that are yet to be filled in terms of SDN security. In order to summarise the evaluative exercise above, a table has been produced, showing each of the solutions, the attack type they aim to defend against, the logical SDN layer they operate at and the security aspects they aim to maintain (table 2).

IV. Discussion and Conclusion

SDN is a promising platform which has been extensively used in research settings due to its highly configurable nature, allowing researchers and experimenters to create bespoke forwarding rules in their environments. It is yet to have made much of an impact in large-scale production settings, due to inherent security issues introduced through the separation of the control and data planes. This paper explored a number of key threats unique to the SDN platform and discusses for each a number of mitigations that have been proposed. It appears that currently one of the weaker areas of SDN security is the link between the application layer, and the network that lies underneath it. Ensuring that this is secure is vital – not only does this link carry sensitive information about the network state, but in production settings, it could potentially carry sensitive client data. By developing robust application frameworks, it would help for a standardised method of developing and deploying applications. This reduces the chance for poor configuration/development and makes it easier to create generic security applications which can be applied to a large majority of SDN deployments.

A general theme which can be observed from the suggested solutions is that they aim to defend against just one particular type of attack, and do not provide overall protection in the network setting. To ensure a secure software-defined network, parallel implementation of combinations of the proposed schemes would have to be considered. This could achieve larger amounts of computational overhead and creates chances of conflicting configurations. These issues lead to the possible introduction of further vulnerabilities to the environment and the chance of false positives when scanning the network for attacks. To provide more thorough and well-rounded security the integration of an intrusion detection system in the controller of the network which identifies and prevents attacks based on pre-defined attack signatures could be implemented. The fact that traffic data is collected in the data plane and passed onto the controller for analysis provides a natural environment to implement IDS. It is clear, however, that with adding security mechanisms such as IDS to the controller, more overhead will be generated and performance of the network in terms of its throughput and the latency achieved could be negatively affected. To improve security in SDN infrastructures, it would be important to find a balance in the trade-off between security and performance by implementing a thorough, network-wide security regime whilst achieving desirable levels of performance. Before implementation of SDN can become wide spread this would need to be researched and developed further.

References

[1] N. Feamster, J. Rexford and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 87-98, 2014.

[2] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "SDN Security: A Survey" in "Future Networks and Services", November 2013.

[3] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, 'Software-defined networking security: pros and cons', IEEE Communications Magazine, vol. 53, no. 6, pp. 73-79, 2015.

[4] I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov, 'Security in Software Defined Networks: A Survey', Communications Surveys & Tutorials, 2015.

[5] D. Kreutz, F. Ramos, P. Esteves Verissimo and C. Esteve Rothenberg, 'Software-Defined Networking: A Comprehensive Survey', Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2015.

[6] Z. Hu, M. Wang, X. Yan, Y. Yin and Z. Luo, 'A Comprehensive Security Architecture for SDN', 18th International Conference on Intelligence in Next Generation Networks, pp. 30-37, 2015.

[7] L. Schehlmann, S. Abt and H. Baier, 'Blessing or curse? Revisiting security aspects of Software-Defined Networking', International Conference on Network and Service Management (CNSM), pp. 382-387, 2014.

[8] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN '13. ACM, August 2013, pp. 55–60.

[9] H. Li, P. Li, S. Guo, and S. Yu, "Byzantine-resilient secure software-defined networks with multiple controllers," in Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014, pp. 695–700.

[10] S. Scott-Hayward, C. Kane, and S. Sezer, "OperationCheckpoint: SDN Application Control," in 22nd IEEE International Conference on Network Protocols (ICNP). IEEE, 2014, pp. 618–623.

[11] J. Hizver, 'Taxonomic Modelling of Security Threats in Software Defined Networking', BlackHat Conference, pp. 1-16, 2015.

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

[13] E. de la Hoz, R. Paez-Reyes, G. Cochrane, I. Marsa-Maestre, J. Moreira Lemus and B. Alarcos, "Detecting and Defeating Advanced Man-In-The-Middle Attacks against TLS", International Conference on Cyber Conflict, pp. 209-221, 2014.

[14] H. Dai, Q. Wang, D. Li and R. Chi-Wing Wong, "On Eavesdropping Attacks in Wireless Sensor Networks with Directonal Antennas", International Journal of Distributed Sensor Networks, pp. 1-13

[15] F. Jiang, C. Song, H. Xun and Z. Xu, "Combat-Sniff: A Comprehensive Countermeasure to Resist Data Plane Eavesdropping in Software-Defined Networks", American Journal of Networks and Communications, vol. 5, no. 2, pp. 27-34, 2016.

[16] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security. ACM, 2013, pp. 413–424.

[17] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in Network Operations and Management Symposium (NOMS), 2012 IEEE. IEEE, 2012, pp. 933–939.

[18] L. Wei and C. Fung, "FlowRanger: A Request Prioritizing Algorithm for Controller DoS Attacks in Software Defined Networks", Next Generation Networking Symposium, pp. 5254-5259, 2015.

[19] N. Dao, J. Park, M. Park and S. Cho, "A feasible method to combat against DDoS attack in SDN Network", International Conference on Information Networking, pp. 309-311, 2015.

[20] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in 19th IEEE International Conference on Network Protocols (ICNP). IEEE, 2011, pp. 7–12.

[21] 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. ACM, 2012, pp. 121–126.

[22] S. Mousavi and M. St-Hilaire, "Early Detection of DDoS Attacks against SDN Controllers", International Conference on Computing, Networking and Communications, pp. 77-81, 2015.

[23] O. O. MM and K. Okamura, “Securing Distributed Control of Software Defined Networks,” in International Journal of Computer Science & Network Security, vol. 13, no. 9, 2013.

[24] O. M. Othman and K. Okamura, "Hybrid Control Model for Flow-Based Networks," in the international conference COMPSAC 2013 - The First IEEE International Workshop on Future Internet Technologies, Kyoto, Japan, 2013.

[25] D. M. F. Mattos, L. H. G. Ferraz, and O. C. M. B. Duarte, “AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking.”, pp. 1-7.

[26] Q. Duan, E. Al-Shaer and H. Jafarian, "Efficient Random Route Mutation considering flow and network constraints", Conference on Communications and Network Security (CNS), pp. 260-268, 2013.

[27] M. Wang, J. Liu, J. Chen, X. Liu and J. Mao, "PERM-GUARD: Authenticating the Validity of Flow Rules in Software Defined Networking", International Conference on Cyber Security and Cloud Computing, pp. 127-133, 2015.

[28] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, “Rosemary: A Robust, Secure, and HighPerformance Network Operating System,” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 78–89.

[29] B. Chandrasekaran and T. Benson, “Tolerating SDN application failures with LegoSDN,” in Proceedings of the 13th ACM Workshop on Hot Topics in Networks. ACM, 2014, p. 22.

[30] P. Zanna, B. O'Neill, P. Radcliffe, S. Hosseini and S. Ul Hoque, 'Adaptive threat management through the integration of IDS into Software Defined Networks', 2014 International Conference and Workshop on the Network of the Future, pp. 1-5, 2014.