Authentication for Resilience: the Case of SDN

Dongting Yu[1], Andrew W. Moore[1], Chris Hall[2], and Ross Anderson[1]

[1] Computer Laboratory, University of Cambridge
[2] Highwayman Associates Ltd.
{Dongting.Yu, Andrew.Moore, Ross.Anderson}@cl.cam.ac.uk,
Chris.Hall@highwayman.com

Abstract. Software Defined Networks (SDN) aim to deconstruct current routers into a small number of controllers, which are general purpose machines, and a large number of switches that contain programmable forwarding engines. The vision is that instead of the ad-hoc mechanisms used in current routers we can build programmable networks using proper computer science abstractions. This technology is now at the startup stage, and is being deployed in the data centres of large web service firms.
We are interested in protecting a future SDN. The current designs follow traditional security assumptions and do not consider many likely deployment scenarios. We discuss how SDN architecture can be structured to offer more security, the auxiliary services that such a network will require and the advantages that it can offer.

1 Introduction

SDN is gaining traction in traditional networking settings by offering a low-cost, programmable alternative to traditional proprietary routers. A large number of hardware switches are controlled by a smaller number of controllers, which are general-purpose computers running special software. This allows network operators to break free from vendor lockin and also holds out the prospect of making networks programmable, leading to the prospect that a number of services currently deployed in proprietary devices (such as firewalls, intrusion detection engines and botnet mitigation) might become applications. The initial deployments are mostly in datacentres where cost savings are paramount [4].

However SDNs will be deployed in less controlled environments too. The question that then arises is what new security problems and protection opportunities may arise in these environments, such as a large airport where 100,000 staff working for 1,000 companies may be sharing the facility owner's network. These airlines, baggage firms, travel agents, catering companies and so on are often competitors and sometimes the agents of states in conflict with each other. A future SDN will have to support good separation between rivals' virtual networks while also supporting dependable shared channels (e.g. of which aircraft is at which gate).

How will the controllers and switches in a complex environment such as an airport authenticate each other? The current standards simply state that SDN

systems may use TLS, but this is not always implemented and would be nowhere near enough. We need to work with or in some cases replace existing network security mechanisms such as MPLS, DNSSEC [1] and BGPSEC [5] while supporting resilience against local failures and service-denial attacks. In an environment like an airport, for example, many switches and some controllers will be on tenant premises and so may be open to occasional compromise. Existing mechanisms not only fail to support resilience; they are incompatible or ill-defined. For example, within an AS, iBGP authentication is not compatible with route reflectors, which cause source IP addresses to not work. (There's a proprietary extension by one router vendor to deal with this, but no standard.) Another example is bandwidth: there are proprietary mechanisms such as Cisco's IP SLA to probe network bandwidth to inform routing decisions, but no standard.

It is therefore time to update the threat model. Just as the traditional protocol research community started off in the days of Needham and Schroder from the assumption that all principals behave themselves, and then had to adapt to cope with misbehaving clients or servers, so also intradomain routing has been traditionally thought to need little authentication which will change as we move to more dynamic networks. The airport with a controller and six switches in a closet that a janitor can access is a very simple case; if and when SDN is deployed on the battlefield, engineers will have to design authentication to cope with devices being constantly added and lost, and occasionally falling into enemy hands.

Scaling also forces a rethink. As we move from current SDN deployments of perhaps 50 controllers and 500 switches in one data centre to global networks with tens of thousands of controllers and hundreds of thousands of switches, we can no longer assume that the threat only comes from outside. It is already an issue that when a network operator deploys a router in a remote location, this is usually done by an untrusted local contractor.

Our threat model must assume physical compromise of devices, along with associated attacks involving (for example) software that's old and vulnerable or that has been tampered with. We must assume that some devices in the field are unsafe; that a handful of switches are compromised at any given time, and sometimes controllers at the bottom of the hierarchy (which are deployed near switches). Some communication channels are also insecure: the wires are subject to the same attacks.

In addition to the usual mechanisms for key generation, distribution, update and revocation, a resilient authentication infrastructure will also require a trustworthy mechanism to monitor and detect rogue devices in the rest of the SDN. This will alert operators when a device starts to act maliciously, so that appropriate action can be taken to revoke and exclude it. The scale and complexity are much larger than previously considered, but a broad range of data can be monitored: we can query a switch, its neighbours and their controllers for bandwidth information; and we can also launch data plane probes to cross-check. With a large corpus of live and historical network data, the operator can make better decisions when under attack.

2 Proposed Architecture

Architecture matters. We can get real benefit from the move from peer routers and switches, any of which can cause equal havoc if compromised, to a hierarchical system of switches and controllers. This means we can arrange things so that the compromise of a few switches will do no more than local damage.

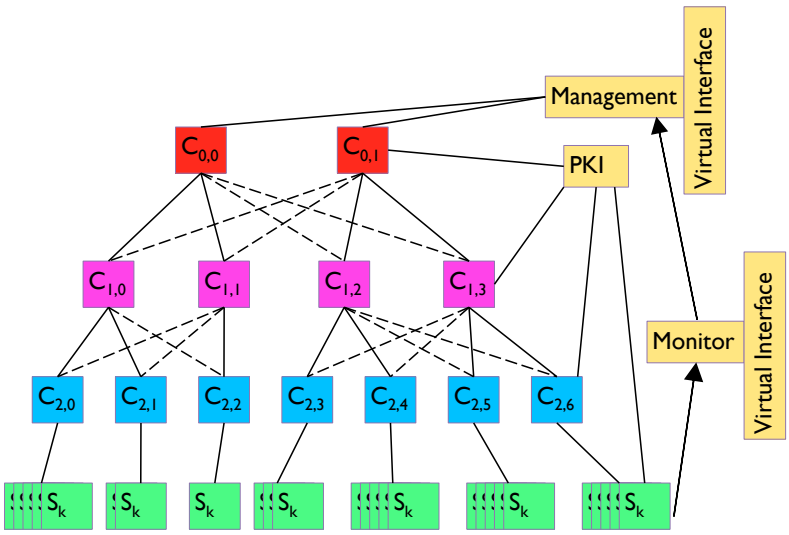

Fig. 1. An SDN setup with hierarchical controllers and switches. Solid lines denote connections, and dotted lines backup connections. Note that the PKI, management, and monitoring services are conceptually drawn, and may not be physically separate from the main hierarchy.

To illustrate this, an SDN currently deployed in a data centre, as illustrated in Figure 1, might have a bottom layer of 1,000 switches, with each ten switches driven by a level 2 controller, every ten level 2 controllers driven by a level 1 device, and the ten level 1 devices coordinated by a master controller. If we can arrange things so that only controllers can cause widespread outages if compromised, the number of critical components is reduced by a factor of ten. If we can further arrange things so that the compromise of a level 2 controller does little damage outside of its immediate neighbourhood, then we have reduced the number of points of serious failure by another order of magnitude.

Although Figure 1 illustrates an SDN hierarchy informed by datacenter practices, without much imagination it is plausible to map the components to those of an ISP (Network Operation Centre, Regional Offices, PoPs, etc.) and to the components of our airport example (where there are some central facilities, some

in separate buildings and some on different floors of those buildings, connected in a hierarchy).

In terms of division of work, the two or more layers of 'middle management' controllers between its root controller and the switch fabric is where the 'work' will be done, of creating virtual networks and supporting virtual services. An operator will issue commands top down from level 0 and each level of controllers below will be responsible for translating the directives into rules suitable for their layer of abstraction. If there is virtualisation, it will also happen in the middle layers, at layers 1 and 2. Finally, the level 2 controllers issue the necessary primitive rules to the switches they control.

We assume the level 0 and level 1 controllers to be trusted, although with potential accidental configuration errors; and that there may be occasional compromises at level 2. However, there may be network application code (the software-defined applications) running on level 1 and level 2 controllers, which might misbehave, intentionally or not. As we noted, some proportion of the switches may be compromised at any one time; and, as the data packets being dealt with can come from anywhere, nothing is assumed of them.

We imagine that in time there will be many SDN applications that operators can choose to deploy. This will bring the same problems seen with application markets for mobile phones. Will we take the 'walled garden' approach of the iPhone, with some central authority that vets applications and developers, or the somewhat more freewheeling approach of Android, where all can play but applications are removed from the play store once they are considered harmful? Many applications will contain too much code to verify, and even if their developers are honest and competent, they may still face commercial incentives to collect as much information as possible, or to give higher priority to their own traffic at the expense of their competitors'. Network engineers deciding how much access to grant an app may be more sophisticated than the typical Android user trying to decide whether a social networking app that asks for the ability to send text messages is exploitative — but the difficulties encountered with the manifests for phone apps bear careful thought. How should we design the set of permissions that will define and constrain the behaviour of an SDN app? How should the access control policies look like? And what will be the practicality if a hundred different virtual networks are run on this fabric for different tenants – will it be at all practical to run different apps on behalf of different tenants on a number of controllers, or will we have to impose significant limitations (such as no shared state for apps on tenant networks, or across controllers)?

We believe this area that needs substantial and urgent research.

3 Auxiliary Services

Apart from the hierarchy of controllers and switches, we will need auxiliary services in the network. Switches are connected to a logical monitoring service which in turn feeds relevant data back into the management service, completing

the loop by connecting to the root level controllers. If TLS is used, there may also be a PKI to support this; an alternative could be a Kerberos-type system.

Monitoring is a logical service in the network. The purpose of monitoring is to collect both control plane status and data plane statistics from the bottom level switches. Monitoring makes available its information to relevant users and operators so they can watch and intervene if needed. This service can perform both passive and active monitoring. Passively, it can measure statistics such as the number of packets matching a certain signature, or per-interface bandwidth usage. Actively, it can send a packet to a switch and observe the decision made by the switch on that packet. Because the monitoring service can observe all interfaces of a switch, it can see the result of a forwarding decision. Monitoring also exposes a new level of control to the network. The potential of using this for auditing and information flow analysis is immense. Among others, SDN makes available an interesting potential for tackling botnet outbreaks as well as adapting and reacting to other forms of network attacks [6, 8].

The monitoring service also feeds data back into the management service. Since this knows all the commands issued from the top, it can check if they are followed, completing the loop. It can can also actively generate fake traffic to isolate devices that are dishonest. Most importantly, it links the human operator with the rest of the network. While the root controllers are the technical authorities in the control plane, the management service translates human operator intentions into control directives.

Both the management service and monitoring service expose a virtual interface for a users of the network, or to those to whom the operator delegates access, for example a network operator for an airline only needs partial access to the airport network. Each such user gets a separate virtual slice of the network along with relevant virtual devices, resources, and monitoring data. On the management side, the virtual interface deals with resource allocation and visibility; and the monitoring side only shows the part of the collected data that the user is authorised to view. There has been some recent work [2, 3, 7] on abstraction and virtualisation in network programming; future research might well focus on incorporating such concepts into the architecture laid out here.

4 Conclusions

Software defined networks are getting deployed because they are cheap, both for hardware capex and for operation. Yet the work on SDN security has only just started. As these technologies escape from the datacenter and get deployed in large heterogeneous networks, a lot of protection issues arise. An important first step is architecture. We propose a hierarchical model of SDN which reduces the number of points of serious failure by one or two orders of magnitude. The significance for protocol research is that while people have in the past talked about hierarchical deployment of both public-key and shared-key protocol mechanisms, this has so far been abstract (and was largely limited to the debate in the 1990s about which cryptographic technology scaled better). For the first time, SDN

provides an environment with a real need for hierarchical security. This in turn raises the question of whether we can use delegation with public key mechanisms, or hierarchical Kerberos mechanisms, to support tiered security in networks.

5 Acknowledgments

We would like to thank Peter Neumann, Phillip Porras, Vinod Yegneswaran, Anil Madhavapeddy, and Charalampos Rotsos for helpful discussions in the early stage of this work. We would also like to thank the audience at the Security Protocols Workshop '13 for additional comments during the presentation.

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), in part under contract FA8750-11-C-0249, and in part under contract FA8750-13-2-0023. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the Air Force Research Laboratory or the Department of Defense.

References

1. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard) (Mar 2005), http://www.ietf.org/rfc/rfc4033.txt
2. Foster, N., Guha, A., Reitblatt, M., Story, A., Freedman, M.J., Katta, N.P., Monsanto, C., Reich, J., Rexford, J., Schlesinger, C., Story, A., Walker, D.: Languages for software-defined networks. Communications Magazine, IEEE 51(2), 128–134 (2013)
3. Gutz, S., Story, A., Schlesinger, C., Foster, N.: Splendid isolation: a slice abstraction for software-defined networks. In: Proceedings of the first workshop on Hot topics in software defined networks. pp. 79–84. HotSDN '12, ACM (2012)
4. Hoelzle, U.: OpenFlow @ Google (2012), keynote address at the Open Network Summit
5. Lepinski (Ed.), M.: BGPSEC Protocol Specification (Feb 2013), http://www.ietf.org/id/draft-ietf-sidr-bgpsec-protocol-07.txt
6. Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G.: A security enforcement kernel for OpenFlow networks. In: Proceedings of the first workshop on Hot topics in software defined networks. pp. 121–126. HotSDN '12, ACM (2012)
7. Reitblatt, M., Foster, N., Rexford, J., Schlesinger, C., Walker, D.: Abstractions for network update. In: Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication. pp. 323–334. SIGCOMM '12, ACM (2012)
8. Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G., Tyson, M.: Fresco: Modular composable security services for software-defined networks. Internet Society NDSS (Feb. 2013). To appear (2013)