The Diamond Approach for SDN Security[1]

Angelo Liguori
Huawei, German Research Center
angelo.liguori@huawei.com

Marcel Winandy
Huawei, German Research Center
marcel.winandy@huawei.com

Introduction

Software Defined Networking (SDN) has attracted a lot of interest in both academia and industry, and it has started to be adopted in real system implementations. SDN emerged as a new concept with the intent of enabling central programmability of the network, although it can be traced back to earlier concepts like Active Networks, Network Control Point, and Routing Control Platform [1,2,3]. SDN is based on the idea of decoupling the control plane from the data plane, introducing a logically centralized control with open interfaces, and providing an API on abstractions of the network elements to program their forwarding behaviour [4]. SDN opens new opportunities for telecommunications and network operators as well as enterprise networks by providing effective means for fast infrastructure provisioning and dynamic reconfiguration of networks.

SDN also poses new challenges to be faced as it introduces new components to the network (APIs, applications, controller). The overall complexity of ensuring security increases, the central control becomes a new popular target, and the openness of interfaces makes it difficult to define and enforce a security policy. Moreover, SDN security has a twofold meaning [5]: “Security by SDN”, i.e. increasing the overall network security of a system using SDN, and “Security for SDN”, i.e., ensuring the secure implementation and operation of the SDN infrastructure itself. We believe that the problem of security for SDN should be addressed first (to a practical and sufficient level) before new security services using SDN can be effectively deployed. A key issue is the security of the SDN controller as it is the “brain” of the network. Any successful attack at the controller can harm the whole network.

Attacks to SDN can take the form of Denial-of-Service, aiming at undermining the availability of network operations, or the form of Man-in-the-Middle with the goal to modify the rules sent to the network devices to take control of the network paths. An attacker would also try to compromise the controller

[1] Original article published at: IEEE Softwarization, March 2018
https://sdn.ieee.org/newsletter/march-2018/the-diamond-approach-for-sdn-security

exploiting vulnerabilities and installing malicious applications, in the attempt of taking full control of the network infrastructure. The mitigation of security risks in SDN requires a secure-by-design approach to provide adequate protection of the infrastructure from both malicious attacks and unintentional vectors (bugged applications, devices misconfiguration, etc.).

Related Work

A couple of secure designs for SDN controllers have been discussed in research, e.g., [6,7,8,9]. FortNOX [9] extends the controller with a conflict detection engine for flow rules and a role-based application authorization. SE-Floodlight [6] adds a security enforcement kernel to the controller, i.e., all operations from applications towards the data plane have to pass a security mediation for authentication and authorization. ROSEMARY [7] focuses on resilience through SDN applications containment, resource utilization monitoring, application permissions, and minimizing functionality in the controller kernel. LegoSDN [8] adds reliability and fault tolerance by providing an application sandboxing design and a mechanism to recover applications from a variety of failures.

While we can find common patterns in these works (separating security from other functionality, isolating applications, etc.), they have not yet been fully adopted by industry. For example, the leading open-source controllers OpenDaylight and ONOS have both some forms of authentication and authorization for external applications, but other security functionality is either lacking (e.g. isolation of internal applications/plug-ins) or mixed together with other functionality in one large executable. A reason for the low adoption might be that the SDN community is missing an overview of the security problems and security patterns and a lack of understanding how to apply them to SDN. In addition, many existing research works neglect some industry demands, e.g. they build their designs on single-instance controllers and do not consider a fully distributed architecture.

The Diamond Approach

In order to overcome these obstacles, we have systemized six core design principles which we consider mandatory for a secure SDN controller architecture. They are derived from best practices of system security and cover common security patterns from the mentioned research works as well as important industry requirements. For illustration, we put these principles as vertexes of a polyhedron, calling to mind the shape of a diamond shielding the controller (see Figure 1). Hence, we call it the *Diamond Approach for SDN Security*.

Complete Mediation: Each time a subject attempts to access a resource, the system should mediate the action. This principle requires systematic access control for resources so that access to them be checked every time to ensure that the subject has proper privileges. The “mediator” should be the logic unique authority for this checking, and it could take advantage of the features provided by distributed systems.

Compartmentalization: This principle, aka Sandboxing, enforces the rule that an occurred security problem should be limited within the specific compartment containing it. This is a well-known concept in all contexts that require safety, e.g. life-critical systems. For SDN it applies to the business and control layers, where applications should be separated and isolated from each other and from the controller itself.

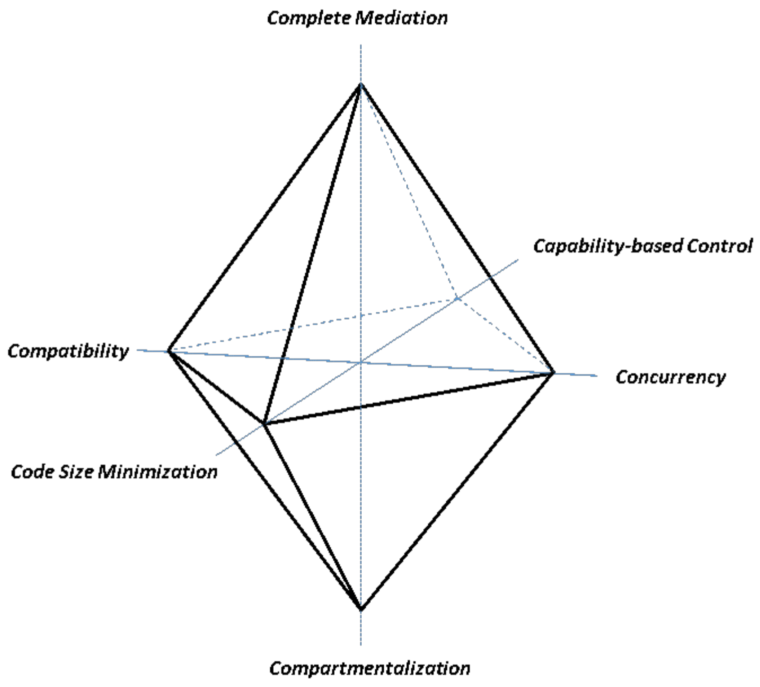

Figure 1: SDN Security Principles (Diamond Approach)

Code Size Minimization: An important aspect in security is to reduce the trusted computing base, i.e., that part of the system that is essentially needed for security and if compromised can totally break the security. First, minimizing the Lines of Code (LOC) reduces the possibility of errors and the attack surface exploitable by an attacker. Second, a thin software layer can be semi-formally or formally verified in order to give mathematical evidences that it correctly fulfils the requirements it claims.

Capability-based Control: This allows to attach authorizations (i.e. capabilities) to the subjects demanding the service instead of attaching them to the resources providing it (Access Control List approach). A capability is completely transferable and it represents the ability to perform privileged operations. In a dynamic environment, where SDN applications and their security context can change often, a capability-based control can be easier managed and audited.

Concurrency: In a distributed system, it is important that components can be executed and work together concurrently. This is extremely important in environments like SDN where controllers can be distributed in clusters and the environment is highly dynamic. Operations like Leader Election and Node Replication are examples where secure concurrency is necessary.

Compatibility: In the variety of SDN controllers currently available on the market, the integration of security mechanisms should not significantly impact the interface design and implementation of existing controllers. A solution to make SDN secure should be transparent for the controller in order to allow customers to apply security in deployed infrastructures and preserve the “openness” SDN originally introduced.

Implementation

We applied the idea of the diamond approach in our Secure Controller Architecture (SCONA) that pools and combines together these principles. Figure 2 depicts an overview of SCONA components, each of one designed to address the SDN security problem in a seamlessly way from the application (business) layer to the data plane.

The main component of SCONA is the *Network Security Kernel (NSK)*. It follows the idea of security kernels from operating systems and is specifically designed to accomplish the principles of complete mediation of all the messages (synchronous and asynchronous) between applications and data plane, compartmentalization of security functionality from the rest of the controller, and code size minimization (our current prototype has less than 20k LOC).

SCONA Controller Applications Sandbox (CAS) and *Controlled Trusted Software (CTS)* also enforce the principle of compartmentalization, avoiding that internal applications could be used to subvert the security of the overall controller and providing an environment for trusted higher-level security functions, e.g. application behaviour monitoring. The remaining, non-security related functionality of the SDN controller is what we call the *Controller Core (CC)* component. SCONA implements a capability-based reference monitor engine that enforces the security check for every application and data plane device through the cooperation of NSK and SCONA Controller Core.

For concurrency, the NSK is also designed to be highly scalable and reconfigurable in order to provide high performance in terms of availability. Finally, NSK is designed to be compatible with existing SDN protocols and controller APIs because it enforces only access decisions between components and their requests. Higher level aspects (e.g., checking consistency of the SDN policy) are left to CTS.

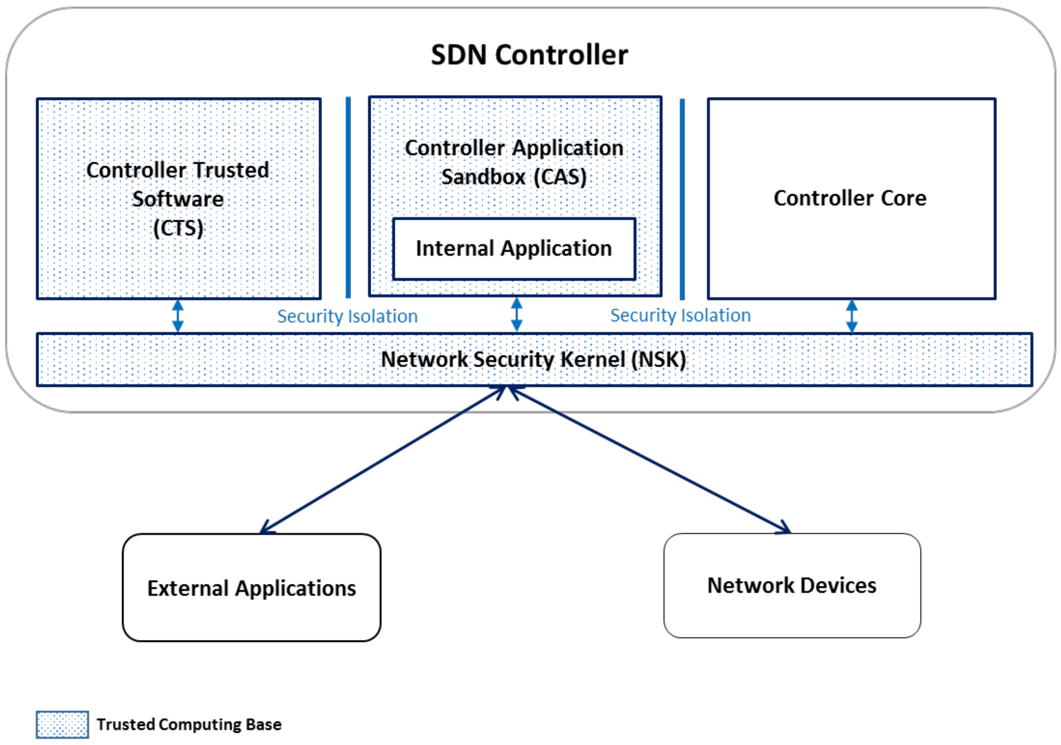

Figure 2: SCONA Architecture

Outlook

SCONA and our diamond approach try to overcome the often partial perspective offered by prior research and existing industry implementations. A key difference and the novelty of our approach lies in the fact that SCONA logically envelopes the controller. It effectively enforces not only the security of applications towards data plane, but also limits the attacks coming from data plane against applications and the controller itself.

Further challenges need to be addressed in order to still enhance the security of SDN. For example a formal analysis of SCONA code or the introduction of a policy checker engine, just to name a few. Formal verification provides evidences of the correctness of the developed components and algorithms, whereas the capability to check if applications' commands could lead to conflicting rules against the established security policies can help maintaining the network in a secure and consistent state. Performance and fault tolerance are also important aspects to address in order to make SCONA scalable and augment the resilience of our solution.

We are currently evaluating and enhancing our prototype according to the afore-mentioned features and hope to push this technology into products in the near future.

References

1. D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, “A survey of active network research” IEEE Communications Magazine, vol. 35, no. 1, pp. 80–86, Jan. 1997
2. D. Sheinbein and R. P. Weber, “800 service using SPC network capability”, Bell Syst. Tech. J., vol. 61, no. 7, pp. 1737–1744, Sep. 1982
3. M. Caesar et al., “Design and implementation of a routing control platform”, Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation,vol. 2, pp. 15–28, 2005
4. M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, W. Kellerer, “Interfaces, Attributes, and Use Cases: A Compass for SDN”, IEEE Communications Magazine, vol. 52, pp. 210-217, 2014
5. S. Scott-Hayward, G. O'Callaghan, and S. Sezer, “SDN Security: A Survey”, IEEE SDN for Future Networks and Services (SDN4FNS), Trento, pp. 1-7, 2013
6. P. Porras, P, S. Cheung, S, M. Fong, K. Skinner, V. Yegneswaran, “Securing the Software Defined Network Control Layer”, 2015.
7. S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. Byunghoon Kang, “Rosemary: A Robust, Secure, and High-performance Network Operating System”. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14). ACM, New York, NY, USA, 78-89. 2014.
8. B. Chandrasekaran, B. Tschaen, and T. Benson, “Isolating and Tolerating SDN Application Failures with LegoSDN”. In Proceedings of the Symposium on SDN Research (SOSR '16). ACM, New York, NY, USA, Article 7, 12 pages, 2016
9. 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 '12). ACM, New York, NY, USA, 121-126, 2012