A Software-Defined Network Honeypot With Geolocation and Analytic Data Collection

Vallie Joseph, Piradon Liengtiraphan, G. Leaden, and Casimer DeCusatis
School of Computer Science and Mathematics
Marist College
Poughkeepsie, NY 12603
casimer.decusatis@marist.edu

Abstract- Software defined networks (SDN) are being widely adopted in cloud computing environments. However, there is a strong need for improved cybersecurity in SDN cloud networks, specifically an approach which protects critical assets such as the SDN controller. We describe a new SDN controller honeypot, known as Dolos, which mimics a real SDN controller and records data on attempted cyberattacks. Experimental data is presented showing that Dolos fingerprints as a real SDN controller. We also present performance testing results while performing mock attacks.

I. Introduction

In recent years, major cloud computing enterprises as well as telecommunication service providers have been adopting software defined networks, including AT&T (via the Domain 2.0 initiative) [1], Verizon, Cisco, IBM, and many others [2]. While conventional data networks use distributed message forwarding, SDN separates the data management and control planes, implementing a centralized, programmable network controller which manages all the routing and flow tables in the network. This approach abstracts key network features and has been shown to provide improved quality of service and energy efficiency while reducing capital and operating expenses compared with traditional approaches [2-4].

Despite the potential advantages of SDN, network security remains a significant concern. Unlike a conventional network, if a centralized SDN controller were compromised, an attacker could gain control of all traffic flows in the network, enabling a wide range of attacks including eavesdropping, denial of service, traffic spoofing, and more [5, 6]. Several new types of cyberattacks have recently been described which specifically target SDN controller [7]. One way to mitigate such threats is by deploying a honeypot designed to mimic the behavior of an actual SDN controller. The use of honeypots to divert cyberattacks from sensitive network components is an effective, well established technique, provided that the attacker cannot easily distinguish the honeypot from a real device [5, 6]. To the best of our knowledge, there are currently no SDN controller honeypots available either from commercial sources or open source code repositories.

We have developed an SDN honeypot known as Dolos (after the Greek god of trickery). Our analysis of Dolos using a variety of reconnaissance tools demonstrates that Dolos fingerprints essentially identical to a real SDN controller from the Linux Foundation's Open Daylight (ODL) project, Lithium release [8]. Furthermore, our honeypot collects useful data on the attacker, such as IP address, duration of the attack, operating system and browser used in the attack, and geolocation information. This data can be processed into actionable threat intelligence. For example, Marist College has independently developed a tool called LongTail to detect attack signatures and classify attacking botnets; this code is available as a Docker container from the IEEE Try-CybSi project, part of the IEEE Cybersecurity Initiative launched by the IEEE Computer Society and the IEEE Future Directions Committee [9].

The remainder of this paper is organized as follows. Section II describes the software design features of the Dolos honeypot and its data collection features. Section III describes a novel geolocation function built into Dolos. Section IV presents results from our fingerprinting comparison of Dolos and an ODL controller. Section V presents the results of our performance testing of Dolos under simulated attacks.

II. Honeypot Software Design

Our initial design concept for Dolos involved deploying a real ODL controller which would not allow successful logins and would collect attack data in a format suitable for the LongTail classifier. This approach had to be abandoned when we found that it was not straightforward to modify the ODL controller to collect attack data, and that some implementations of ODL showed poor performance under high traffic load. Instead, we created a new honeypot to address these concerns. Dolos was written in PHP, and currently runs on Apache web server version 4.2.7 (Ubuntu). We chose to implement Dolos in PHP/MySQL version 5.5 to provide additional flexibility in our data collection, and to avoid several documented issues with the current ODL Java release [8]. The Apache web server was chosen in an effort to insure robust performance under high volumes of attack traffic, such that our honeypot would continue collecting data under conditions which might otherwise overwhelm a real ODL controller. Our implementation also takes advantage of several existing Apache Server modules (mod rewrite and mod security) which include specialized Linux files, Angular JS directives, and selected custom JavaScript functions [10]. In order for Dolos to be effective, it should fingerprint as a real ODL controller. Otherwise, an attacker might be able to determine that Dolos is a honeypot, and bypass it altogether. ODL is a Java-based SDN controller, and uses a version of Java servers known as Jetty [11]. Since our honeypot is not based on Java, it was necessary for the honeypot to spoof key features of ODL.

One example is the graphic user interface (GUI) which ODL uses to prompt for userid and password credentials. Although SDN attack vectors have not yet been well characterized in the literature, we anticipate an exhaustive dictionary attack attempting to brute force the login credentials, similar to existing SSH attacks [5, 9]. While Dolos also presents a login prompt, ODL authenticates each login using a java function, which Dolos does not possess. We replaced the java function with a PHP form submission in Dolos, which subsequently retrieves attacker information. This difference is not easily detected by most cursory reconnaissance scans of Dolos, however we recognize that more sophisticated attackers may possess a multitude of specialized tools for port scanning, vulnerability scanning, directory traversal, and much more. Should the attacker perform a detailed scan or "web crawling", they may be able to determine not only each page the browser is calling, but the file extension types as well. This would make it possible to determine what languages are being used on the page.

The most effective solution for masking the usage of PHP on the server would be to "rewrite" the login page to resemble a page that is already being called with the ODL controller. After login credentials are submitted, the ODL controller calls a file named "modules" within the "restconf" directory. Typically, this file is used to display virtual router data for ODL. However, since the honeypot does not actually connect to any other networking equipment, there is no need to install and configure virtual routers; the honeypot will still resemble a real controller. Instead, a combination of Linux's symbolic links (also known as symlink or softlink [12]) and the mod rewrite from Apache 2 are employed. Using mod rewrite, we enter a Regular Expression that accepts two parameters: the fake page we want the attacker to see, alongside the real page that we wish to hide. After mod rewrite has been applied, we create a symbolic link to the rewritten page within the login script. For example, if the usual login script calls for a page named "login.php", we would use mod rewrite to make this appear as a non-php page of a different name. The "login.php" page would then display as whatever name and extension we choose, for example "foo.txt". Following the mod rewrite command, we then create a symbolic link to artificially link "login.php" and "foo.txt" together in such a way that by calling the "foo.txt" file, we are actually calling the "login.php" file. In this way, we are able to spoof both the file extension and name to a pre-existing file/file type within the real controller to enhance the honeypot's authenticity.

Furthermore, we used an Apache security module known as mod security to rewrite and spoof the response/request header output that would be viewable to an attacker. For example, when utilizing a server response scanner such as Nikto or Sparta (as discussed in section IV), the attacker would typically be able to see objects such as request methods, the page status code, the connection type which the server maintains, the user agent of the attacker, and the name of the server. We focused on masking the server name as the same Jetty server name located on the ODL controller, using the configuration files that are packed within mod security. Here, we are able to edit a field named "SecServersSignature" and set the value to the desired value, in our case "Jetty(8.1.15v20140411)".

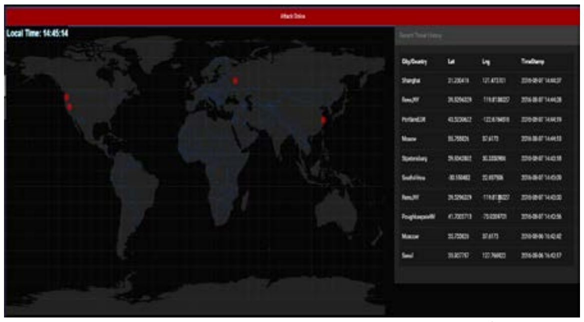
Prior attempts to develop honeypots for SSH and other protocols [5, 6, 8] typically collect data such as source and destination port/IP address, time of day, and attempted login credentials. Dolos takes a similar approach and collects this data while never allowing a successful login. We also decided to collect data on the attacker's operating system and browser, which may provide useful in identifying and classifying attack patterns. To achieve this, we extracted information from the User-Agent, a property that is automatically sent by the browser which an attacker is using against the honeypot. As the attacker attempts to gain access, their user agent is recorded and parsed using several javascript functions. Dolos is able to determine nearly any type of operating system (from mobile devices to different Linux distributions), to the exact type and version number of the browser used in the attack.

III. Geolocation using Dolos

One of the novel features built into Dolos is the additional collection of Geolocation and ISP data using free, open source resources. We have written an open source Python script, known as GeoCheese, which provides information on the attacker including geolocation data (country, subdivision, city, postal code, latitude, and longitude) and ISP (hostname, company name, ASN, and host IP). Dolos executes GeoCheese as a system command. Originally, this module was imported by another standalone Python script which handled scraping the syslog files and then inserting into a database; this functionality was later subsumed into Dolos.

```
Target IP: 96.245.176.198
Retrieving basic geolocation data from GeoLite2-City.mmdb…
Successfully retrieved ISP data from WhatIsMyIPAddress…
Successfully obtained GeoISP Data – 49.287.66.124 {'city':
'Secunderabad', 'host': 'broadband.actcorp.in' 'subdivision':
'Telangana', 'name': 'Beam Telecom' 'ip': '49.207.66.124' 'lat':
'17.45', 'long': 78.5' 'country': 'India' 'postal': '500087' 'ASN':
'55577'
```

Figure 1 – Screen shot of sample code illustrating Dolos geolocation and results



City/Country	Lat	Lng	TimeStamp
Shanghai	31.230416	121.473701	[illegible]
Reno,NV	39.5296329	-119.8138027	[illegible]
Portland,OR	45.5230622	-122.6764816	[illegible]
Moscow	55.755826	37.6173	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
SouthAfrica	-30.559482	22.937506	[illegible]
Reno,NV	39.5296329	-119.8138027	[illegible]
Poughkeepsie,NY	41.7003713	-73.9209701	[illegible]
Moscow	55.755826	37.6173	[illegible]
Seoul	[illegible]	[illegible]	[illegible]

Figure 2 – Sample Dolos GUI visualizing geolocation information from global attack data

Figure 1 illustrates sample code showing the primary sources of information concerning each IP address provided for Geolocation and ISP, respectively. We also created a GUI shown in Figure 2 which visualizes the geolocation results. Geolocation is a multi-step process, to ensure the completeness of the data returned. First, we query the IP address against the open source database Maxmind GeoLite2-City.mmdb [13]. Second, we pass the IP address through another Python module called Geocoder, which attempts to retrieve additional information not provided by the first query. Third, we pass the latitude and longitude provided by GeoLite2 into the online Google Maps API and the data is retrieved via our GeoCheese application. We also attempt to retrieve data on the attacker's ISP using an open source tool called "What's My ISP" [14]. Packaged with the GeoCheese module are two separate databases/reference files and the Python module Pyasn, which are used for IP to ASN lookup and ASN to Name lookup, respectively. This is used in tandem with another online source "Who's My ISP" [14].

IV. Experimental Fingerprinting of Dolos

In order to demonstrate that Dolos effectively spoofs an ODL controller, we performed a series of scans comparing the two controllers. In order to get a well-rounded idea of what an attacker would see when attempting to compare Dolos with a real ODL controller, we used a variety of popular scanning tools included with Kali Linux [15], including web server scanners (Nikto), FTP and remote file vulnerability scanners (Uniscan), HTTP directory traversal scanners (DotDotPwn), general data scanners (Enum4Linux) and finally a port and trigger identifier (Amap). Each scan was attempted multiple times to insure consistency of the results.

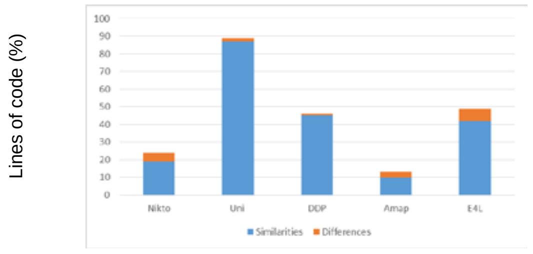


Figure 3 – Percentage lines of code which were identical (blue bar) and different (orange bar) between Dolos and ODL controller, using different scanning tools

A summary of the scan results is shown in figure 3, comparing lines of code which differed between Dolos and the ODL controller (different scan tools evaluate the code in more or less detail, so not all scans show the same total lines of code). As shown in the figure, these scans found only very slight differences between Dolos and the ODL controller (less than 7 lines of code worst case, for scanning between 10-50 total lines of code). Some of these are attributed to details such as the actual address where the honeypot and real controller reside, information which would not be helpful to an attacker trying to determine if Dolos was a real controller. Any remaining differences which might suggest the controller is a honeypot are only revealed when using a relatively slow, highly detailed scan to isolate port details, which we feel is an acceptably low risk. While no honeypot is foolproof, based on these results we have high confidence that attackers will not easily be able to determine that Dolos is a honeypot.

V. Performance Testing

After insuring that Dolos can successfully spoof a real ODL controller under the vast majority of conditions, it was important to demonstrate that Dolos can also withstand large scale attacks within relatively short windows of time. In order to test this, we developed a web application that connects with a tool named

ApacheBenchmarking [16]. This tool runs benchmark tests with user-specified parameters such as total number of users, number of concurrent users, server address, and desired port on which to access the server. We conducted an extensive series of tests with different combinations of these parameters, such as testing different concurrent users on different ports or testing how the server handles more total users whilst having a low concurrency level. Representative results are shown in Figure 4, which shows server response time vs number of requests for different fractions of concurrent users.

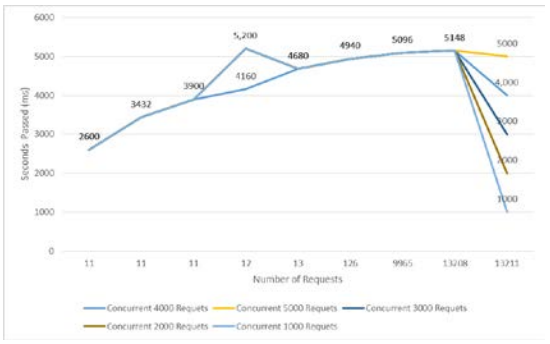


Figure 4 – Representative results of performance testing; honeypot Apache server response time vs number of requests, for different numbers of concurrent requests

Based on existing benchmark data [17], we expect that attacks on the honeypot will continue as long as the response time to a failed login request is less than 10 ms. As seen in Figure 4, our worst case response time was 5.2 ms. We have not yet attempted any performance tuning, which would be expected to further improve the server response.

Summary and Conclusions

We have developed and tested a novel SDN controller honeypot called Dolos, which fingerprints as virtually identical to a real Open Daylight controller. Performance testing shows that the honeypot would be able to withstand expected traffic volumes associated with a brute force dictionary attack on the controller access credentials. Our honeypot acquires data from the attacker, including geolocation information. Future work includes deploying Dolos in our production data center, and parsing this data into actionable threat intelligence using LongTail.

Acknowledgements

The authors wish to thank the Marist/IBM Joint Study, and the Marist College IT group. This research has been supported in part by NSF grant 15-534, Campus Cyberinfrastructure – Data, Networking, and Innovation Program (CC*DNI), aka SecureCloud.

REFERENCES

[1] AT&T Domain 2.0 Vision White Paper, November 2013 https://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf (last accessed May 20, 2016)

[2] S. Das, D. Talayco, and R. Sherwood, "Software defined networking and OpenFlow", Chapter 17 (pp. 427-444) in Handbook of Fiber Optic Data Communication, 4th edition (C. DeCusatis, editor), Academic Press, NY (2014)

[3] C. DeCusatis, R. Cannistra, and L. Hazard, "Managing multi-tenant services for software-defined cloud data center networks", Proc. 6th annual IEEE international conference on Adaptive Science & Technology (ICAST 2014), Covenant University, Nigeria (October 29-31, 2014)

[4] C. DeCusatis, R. Cannistra, B. Carle, "A demonstration of energy efficient optical networks for cloud computing using software-defined provisioning", Proc. IEEE GreenCom (November 12-14, 2014) http://www.ieee-onlinegreencomm.org/program.html (last accessed Sept. 2016)

[5] R. Smith, Elementary Information Security, 2nd edition, Jones and Bartlett, Burlington, MA (2016)

[6] S. Oriyano, Hacker Techniques, Tools, and Incident Handling, 2nd edition, Jones and Bartlett, Burlington, MA (2014)

[7] M. Conti, F. DeGaspari, and L. Mancini, "Know your enemy: stealth configuration information gathering in SDN", see http://www.theregister.co.uk/2016/08/23/sdns_normal_behaviour_is_sniffable_say_researchers/ (posted August 2016, last accessed Sept 2016)

[8] Linux Foundation documentation on Open Daylight, available from https://www.opendaylight.org/ (last accessed Sept. 1, 2016

[9] E. Wedaa, "LongTail: a brute force SSH analyzer", Proc. Derbycon 2015, Louisville, KY (2015); see also IEEE TryCybSi Project, LongTail http://try.cybersecurity.ieee.org/trycybsi/explore/honeypot (posted March 2016, last accessed Sept. 2016)

[10] Apache Server documentation, available from https://httpd.apache.org/ (last accessed Sept. 1, 2016)

[11] Jetty Server documentation, available from http://www.eclipse.org/jetty/ (last accessed Sept. 1, 2016)

[12] Usenix documentation, available from https://www.usenix.org/ (last accessed Sept. 2016)

[13] MaxMind documentation, available from http://www.maxmind.com (last accessed Sept. 2016)

[14] Geolocation documentation includes https://whatismyip.com, https://www.whoismyisp.org ,http://www.w3schools.com/jsref/prop_nav_useragent.asp,http://man7.org/linux/manages/man2/symlink.2.html (all last accessed Sept. 2016)

[15] Kali Linux documentation, available from https://www.kali.org/ (last accessed Sept. 1, 2016); see also D. Dieterle, *Basic Security Testing with Kali Linux 2*, CreateSpace (2016)

[16] Apache Benchmark tool documentation, http://httpd.apache.org/docs/current/programs/ab.html (last accessed Sept. 2016)

[17] J. Nielsen, "Website response times", a report from Nielsen Norman consulting group, https://www.nngroup.com/articles/website-response-times/ (posted June 2010, last accessed Sept. 2016)