Autonomic Security for Zero Trust Networks

Dayna Eidle, Si Ya Ni,
and Casimer DeCusatis
Marist College
Poughkeepsie, NY 12601

Anthony Sager
BlackRidge Technology
Reno, NV 89521

***Abstract*—There is a long-standing need for improved cybersecurity through automation of attack signature detection, classification, and response. In this paper, we present experimental test bed results from an implementation of autonomic control plane feedback based on the Observe, Orient, Decide, Act (OODA) framework. This test bed modeled the building blocks for a proposed zero trust cloud data center network. We present test results of trials in which identity management with automated threat response and packet-based authentication were combined with dynamic management of eight distinct network trust levels. The log parsing and orchestration software we created work alongside open source log management tools to coordinate and integrate threat response from firewalls, authentication gateways, and other network devices. Threat response times are measured and shown to be a significant improvement over conventional methods.**

I. Introduction

In recent years, both the number and severity of cybersecurity incidents have grown dramatically [1], [2]. To cite just a few examples, in May of 2016 an attack on the global financial messaging network known as SWIFT resulted in over $80 million in losses to banks in Bangladesh, the Philippines, Sri Lanka, Vietnam, and elsewhere [2]. This attack highlighted the ongoing need for protection against insider threats from within a presumably trusted network and a lack of rigorous authentication and identity management for network users. Many other incidents have demonstrated a need for improved defense against automated cyberattacks, including botnets and crypto-ransomware. In October 2016, several notable distributed denial-of-service (DDoS) attacks occurred, harnessing botnets comprised of security cameras, baby monitors, home appliances, and other devices on the Internet of Things (IoT). The Mirai botnet disrupted DNS servers at the service provider Dyn, affecting millions of users on Twitter, Amazon, Spotify, Netflix, Tumblr, and Reddit [1], [2]. When the Mirai source code was released shortly thereafter, variants reportedly disrupted banks and telecom carriers in Russia, Germany, and the United Kingdom. In early 2017, a family of crypto-ransomware and disk wipers called Petya disabled major hospitals in the United Kingdom, forcing ambulances to be re-routed and impacting operating theatre equipment [1], [2]. These examples are part of an unfortunate rising trend in cyberattacks, which motivate an urgent need for improved defensive capabilities.

This work was supported by the National Science Foundation under CC*DNI Integregation (Area 4): Application Aware Software-Defined Networks for Secure Cloud Services (SecureCloud). Award #1541384.

Even more concerning, the cyber-defense capabilities of many organizations are already stretched to their limits. According to the 2017 Security Capabilities Benchmark Study [1], most organizations can only investigate about half of the alerts they receive on any given day. This is at least partly due to the massive number of potential security alerts that must be processed; 44% of security operations managers receive over 5,000 security alerts per day [1]. In this environment, less than half (about 46%) of legitimate security alerts are actually addressed, while the remainder are left uninvestigated [1], [2]. Investigation of a security alert is currently a manually intensive process, which can take several minutes to hours or longer [2]. Further, the impact of failing to detect these attacks can be quite significant; 30% result in lost revenue and 4% result in lost business opportunities [1].

There is a corresponding need to improve both efficiency and response time to immediate threats; automated systems are good candidates for this approach. Further, nearly half of network security breaches have impacts lasting between several hours and a full day [1], partly because there is a lack of coordinated follow-up activity to an initial security alert. Many studies cite a need for increased automation in security architectures to reduce the response time and improve the effectiveness of cybersecurity defenses [1], [2], [3], [4]. Finally, another contributing factor is the lack of integrated security tools, which can create gaps in the response time of conventional cyber defenses. A majority of organizations (55-65%) use six or more different security products from just as many different vendors. Hampered by the number of different network components acting in an uncoordinated manner, many organizations fail to mount an effective defense.

A number of security architectures have been proposed to address these concerns. The National Institute of Standards and Technology (NIST) has proposed the Zero Trust Model [5], which has recently been reaffirmed by a 2016 report from the U.S. House of Representatives Committee on Oversight and Government Reform [6]. The Zero Trust Model does not assume trust for any entity (including users, devices, applications, and packets), regardless of whether such entities are inside or outside a secure network. This approach is based on three key principles. First, it eliminates trusted zones within the architecture; all resources must be accessed securely, regardless of location. Second, access control policies are strictly enforced, sometimes at multiple locations in the design including gateways and firewalls (a strict least privilege approach). Third, all network traffic should be logged, inspected, and analyzed regardless of its origin. These fundamentals are closely aligned with the more recent NIST concept of Continuous Diagnostics and Mitigation [7], which builds upon the concept of near real time analysis of all network traffic and transactions. This represents a dramatic change from traditional security approaches. The processing requirements for security telemetry data, such as network equipment logs, a major research challenge.

While zero trust networks remain largely a theoretical abstraction, some progress has been made towards achieving their goals. One approach to continuous diagnostics and mitigation involves control plane feedback loops, which performs rapid threat analysis close to the point where a threat is detected, followed by a more elaborate considered response to prevent long-term damage. Such an approach has been proposed previously, for example, Boyds OODA (observe, orient, decide, act) model [8]. Boyds original work and subsequent research in this area [4], [5], [8] has noted the importance of developing a defensive control loop response which operates at a faster tempo than the attackers ability to respond. In cybersecurity applications, this requires extracting actionable intelligence by parsing large data sets to isolate specific attack signatures on a time scale of minutes or less (depending on the type of attack). Such an approach allows a defensive system to avoid storing huge quantities of raw data in real time. Instead of storing all the system log data, it is preferable to parse the logs and only use a small fraction of telemetry data, which provides useful information about the current situation. Another useful approach is derived from autonomic computing, recognizing that human security administrators cannot keep pace with the volume, velocity, and variety of security alerts and related data [3]. Analogous to various functions in human biology, autonomic computing refers to the self-managing characteristics of distributed autonomous resources, adapting to unpredictable changes in near real time while hiding intrinsic complexity [4]. Increased automation of cyber-defense leveraging OODA and autonomic principles holds potential for defending cloud and enterprise data center networks. It may also help address the current significant lack of trained cybersecurity professionals [1] by automating some fraction of common security tasks.

In this paper, we present results from an experimental cybersecurity test bed, which implements aspects of a zero trust data communication network using autonomic OODA loops. We present test results of trials in which identity management with automated threat response and packet-based authentication were combined with dynamic management of eight distinct network trust levels. The log parsing and orchestration software we created work alongside open source log management tools to coordinate and integrate threat response from firewalls, authentication gateways, and other network devices. We coordinate and integrate threat response from multiple network devices, including an authentication gateway from BlackRidge Technologies, and the Juniper SRX firewall security platform. This approach streamlines the process of detecting and mitigating threats and provides a more efficient response to DDoS attacks and other major threat use cases.

The rest of this paper is organized as follows. After an introduction to the problem in section I, we describe each of the autonomous network components and our approach to zero trust authentication in section II. In section III, we describe the cloud computing network security test bed, including our original software, test results when conducting mock attacks on our network, and analysis. Finally, section IV summarizes our conclusions and potential objectives for future work.

II. Autonomous Network Components

Our test bed is shown in Fig.1, including both the data plane (thin arrows) and control plane (wide arrows). Conventional network security architectures place firewalls between untrusted networks (such as the Internet) and protected resources on an internal network, as illustrated by the data plane in Fig. 1. By contrast, our test bed implements dynamic orchestration of firewall access control lists (ACLs). We also incorporate authentication gateways, which have their own dynamic trust level configuration. The authentication appliances implement First Packet Authentication (FPA) and Transport Access Control (TAC). In principle, any network equipment with an API (preferably a RESTful one) can be used in this architecture. In the following sections, we briefly describe the components used in our test and some of their novel features and then discuss our experimental results.

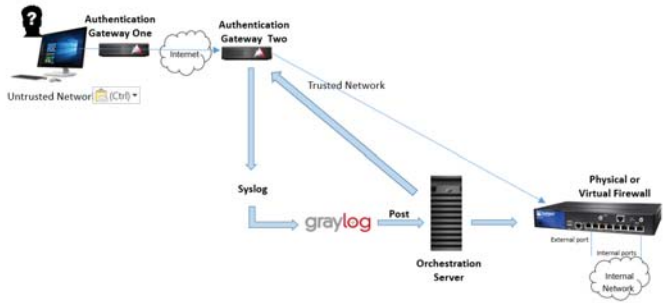

Fig. 1. Experimental test bed.

A. Authentication Appliances with FPA, TAC, and Dynamic Trust Levels

In our proposed explicit zero trust network architecture, authentication and access control are provided through a novel combination of two technologies mentioned earlier, FPA and TAC[9]. These techniques are implemented through the BlackRidge authentication gateways running code release 3.1 [9]. Explicit trust is established by generating a network identity token during session setup. The network token is a 32-bit, cryptographically secure, single use object (with a 4-second lifetime). A token includes an entanglement of the session ID, users IP address, and other information as discussed in prior references [9]. Tokens are associated with identities from existing Identity Access Management (IAM) systems and credentials, such as Microsoft Active Directory or the IAM system used by Amazon Web Services [10]. Trusted users have identity tokens inserted by Gateway One; untrusted users receive no such authentication tokens. Explicit trust is established by authenticating these identity tokens on the first packet of a TCP connection request, before the conventional 3-way Ethernet handshake is completed and before sessions with cloud or network resources are established. Tokens are generated for each unique entity (with a static identity defined on the gateway) requesting access to a network resource; these entities are generally a user or device. As shown in Fig. 1, a pair of inline security gateways (either physical or virtual) are deployed at either edge of an untrusted network. The first gateway inserts the identity token while the second gateway attempts to authenticate the token. When the second gateway receives a connection request, it extracts and authenticates the inserted identity token and then applies a security policy (such as forward, redirect, or discard) to the connection request based on the received identity. Each network session is independently authenticated before any access to the network or protected servers is granted. Unauthorized traffic is simply rejected from the network, and there is no feedback from the transport layer or below to a potential attacker attempting to fingerprint the system. This implementation of TAC ensures that resources behind the second gateway are hidden from network scanning attempts, and only authorized identities are passed on to the network firewall and protected resources [9]. All attempted access is logged to an external Syslog server, which allocates enough memory to avoid wrapping and overwriting log entries. We emphasize that continuous logging of all access attempts is consistent with the approach of a zero trust network (i.e. not allowing any access attempts to go unmonitored). We also note that a conventional intrusion prevention system (IPS) is no longer required with this approach.

Our implementation of the gateways employs eight defined system trust levels exposed through the gateway API, as shown in Table 1. The lower the trust level, the more restricted the users actions become. For example, trust level 0 blocks all traffic, while Trust level 7 allows all traffic. Trust levels 3 through 6 are reserved for definition of application or client specific security policies, which may be enforced using one of three options, namely Global, Group and Identity. The Global trust level will apply to all traffic through the Gateway. The Group trust level will apply to collections of identities, which may be configured on the Gateway. The Identity trust level will apply to traffic related to individual identities. The trust level for a given session is determined to be the most restrictive of the Global, Group, and Identity levels. Trust levels 0 through 2 are reserved for defining system wide policies which override any customer defined policies at trust levels 3-6; the default actions for these trust levels can be changed by ONLY the System Administrator. Attempting to set a trust level without a corresponding policy definition is not permitted; the attempt to do so; will result in the packet being assigned the next lower and more restrictive trust level. Trust level will remain constant for a given TCP session from connection to disconnection but may change on subsequent sessions between the same source and destination if any of the trust levels have been modified. Trust level is set on the resolving gateway, it is not propagated with an identity.

TABLE I.
ALLOWED TRUST LEVELS FOR THE AUTHENTICATION GATEWAY

Level 7	Least restrictive; by default, forward all traffic on trusted and untrusted interfaces (note: requires a configured route table or NAT table to operate properly in some cases)
Level 6	Customer Policy, Group Level
Level 5	Customer Policy, Group Level
Level 4	Customer Policy, Group Level
Level 3	Customer Policy, Group Level
Level 2	System Wide policy defined by admin only
Level 1	System Wide policy defined by admin only
Level 0	Most restrictive, System Wide policy; blocks all traffic on trusted and untrusted interfaces

This approach has several advantages, including separation of security policy from the network design (i.e. network addresses and topologies) [9], [11]. It works for any network topology or addressing scheme, including IPv4, IPv6, and networks which use the Network Address Translation (NAT) protocol; it is also compatible with dynamic addressing often used with mobile devices. This method extracts, authenticates, and applies policy to the connection requests, not only protecting against unauthorized external reconnaissance of the network devices but also stopping any malware within the protected devices from calling home (exfiltration). Security policies can be easily applied at the earliest possible time (on the first packet of a network session request) to conceal network attached devices from unauthorized awareness. By preventing unauthorized scanning and reconnaissance, TAC disrupts the attackers kill chain, blocks both known and unknown attack vectors, and stops lateral attack spreading within a data center. This approach is low latency and high bandwidth since packet content is not inspected. Since the network tokens are embedded in the TCP session request, they do not consume otherwise useful data bandwidth. The combination of TAC and firewalls to produce a segmented, multi-tenant network implements a layered defense against cybersecurity threats and contributes to non-repudiation of archival log data. These techniques are also well suited to protecting public and hybrid cloud resources or valuable, high performance cloud resources such as enterprise-class mainframe computers. Further, as discussed in prior work [9], this approach can be applied to software defined networks (SDN), protecting the centralized SDN network controller from unauthorized access, and enabling only authorized SDN controllers to manage and configure the underlying network. Our implementation of TAC uses an innovative identity token cache to provide high scalability and low, deterministic latency. The token cache is tolerant of packet loss and enables TAC deployments in low bandwidth and high packet loss environments. Functionality and basic performance of FPA and TAC have been established previously [9], [12].

B. *Firewall with Dynamic ACL Control*

Our test bed demonstrated dynamic ACL control using a commercial physical firewall, the Juniper SRX210 Services Gateway [13], which is enabled with a RESTful API used by our control plane software. This platform is designed to provide firewall services for small and medium sized businesses, enterprise branch and remote offices, and certain types of cloud service provider environments. While there are many products in the SRX platform which are potentially compatible with our approach, we selected the model 210 due to its cost and availability to demonstrate autonomic control in our research test bed. This device supports LAN and WAN connectivity and provides additional features including support for Internet Protocol Security (IPsec), virtual private networks (VPN), and Unified Threat Management (UTM), which consists of IPS antispam, antivirus, and web filtering. It also offers optional features through the most recent release of the Junos operating system, with no additional hardware required [13]. While our test bed only requires the REST API, this platform is compatible with SDN controllers, which could also be used to dynamically control firewall properties in our implementation.

III. Experimental Test Bed Results

The experimental test bed used to investigate autonomic security was shown previously in Fig. 1, including both the data and control planes. The OODA control loop is implemented as follows. We implement observation of network events through the authentication gateway logs. We orient our analysis to a specific attack signature. The decide step and act step are first implemented within the authentication gateway, quickly placing suspect attackers on a short duration blacklist. This initial implementation of an OODA loop is supplemented by a second loop which adjusts the firewall ACL. In this case, observation is still based on the authentication gateway logs, while orientation is accomplished through the Graylog tool and our log parsing code. The decision to make changes is made by our policy code in the orchestration server, and the action is implemented through API calls to the firewall. These two OODA loops allow us to detect and initiate an autonomic response to DDoS attacks.

In order for users to access secured assets on this network, the first packet of their connection request is authenticated using a pair of BlackRidge ver 3.1 network security gateways. The first gateway creates and inserts a time sensitive identity token (valid for four seconds) into the initial TCP/IP connection request. After traversing an untrusted network (such as the Internet) the identity token is authenticated by a second gateway. Unauthorized packets are discarded, and TAC prevents any further feedback which an attacker might use for reconnaissance. Once the session has been authenticated, a firewall is used to filter traffic before it reaches the protected resources; this provides an additional point of management control and supports a defense-in-depth strategy.

For example, the BlackRidge gateway detects a DDoS attack whenever there are more than 100 unsuccessful access attempts against a given network address in a 60-second interval (the time interval and number of events are configurable parameters). Upon detecting this event, the authentication gateway can be programmed to deny subsequent authentication attempts from untrusted users, for a period ranging from 30 seconds to several hours. Essentially, there is a control plane feedback loop inside the BlackRidge gateway itself, which effectively blocks the attack immediately (no unauthorized packets can access the network). The worst case latency for blacklisting an IP address within the BlackRidge appliance is under one second and typically on the order of tens to hundreds of milliseconds.

Further, it may be desirable to dynamically adjust other network equipment depending on the nature of the attack. For example, in the case of very large DDoS attacks, or DDoS attacks combined with the insertion of malware while network manager visibility is impaired [13], it is desirable to also update the firewalls ACLs which may be impacted by such an attack. In our test bed, detection of a DDoS event creates a time-stamped alert in the BlackRidge logs which includes the source IP address. The attempted access is logged to an external Syslog server, which allocates enough memory to avoid wrapping and overwriting log entries. We note that continuous logging of all access attempts is consistent with the approach of a zero trust network. The open source logging tool, Graylog, [14] detects this event and triggers our log parsing script with an HTTP alarm callback. Optionally, we can pass the IP address as an argument to a device-specific API. To control trust levels on the BlackRidge authentication gateway, we used the API built into version 3.1 of the management software. In the case of the Juniper SRX, we used the open source Junos PyEZ library [13] to configure connections with the firewall services gateway. This creates a closed-loop feedback in the network control plane, automatically adjusting the network security parameters in response to the attack.

We measured the latency for dynamic orchestration of firewall ACLs in response to mock DDoS attacks using Wireshark version 2.2.7 running in the orchestration server shown in Fig. 1. The response time is measured from the moment when the BlackRidge gateway detects a DDoS attack and logs the IP address in its blacklist until the moment when the orchestration server calls the firewall API and successfully adds the attacking IP address to the firewall blacklist.

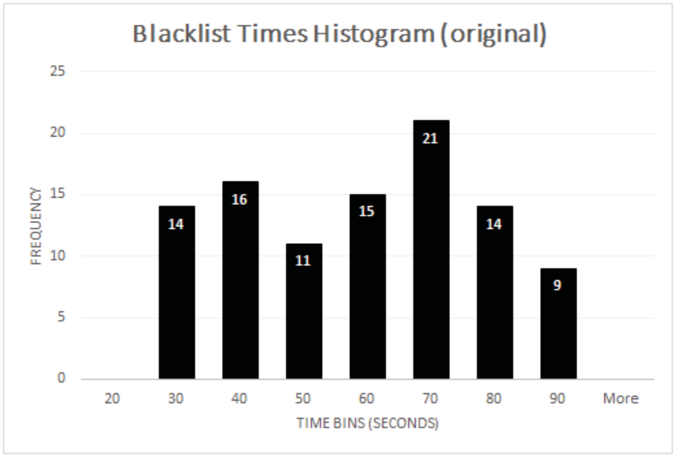

Fig. 2. Frequency distribution of OODA loop response times, 10-second intervals.

The distribution of response times for 100 trials is shown in Fig. 2, binned into 10-second increments. Response time varied from a minimum of 27-seconds to a maximum of 84-seconds. This is significantly faster than the response time of a human systems administrator, and fast enough to prevent our authentication appliance from succumbing to the DDoS attack [9], [10], [11], [12], [13]. We also measured the factors contributing to total latency by randomly sampling data points from each of the bins used in the previous figure. For each point, we measured two time intervals, namely the time between the gateway detecting a DDoS attack and the time when this attack was posted to the log (the add-to-post time), and the time between the attack being posted to the log and the firewall rule update completion (the post to finish time). Results showed that the add-to-post time was the largest component of total latency. The post-to-finish time was very consistent (varying between 17-24 seconds) while the add-to-post time varied considerably (between 4-59 seconds). We tried to reconcile that discrepancy and discovered it was caused by GrayLog triggering alerts once every 60 seconds. The add-to-post times varied depending on when the attack occurs relative to this export. This suggests that it should be possible to significantly reduce total latency by modifying GrayLog to export events as soon as they are detected. Further, the frequency of exports might be adjusted depending on whether an attack was recently detected. We modified the GrayLog reporting interval to 10 seconds, and were able to significantly reduce response time (from a range of 27 to 84 seconds to one of 21 to 32 seconds).

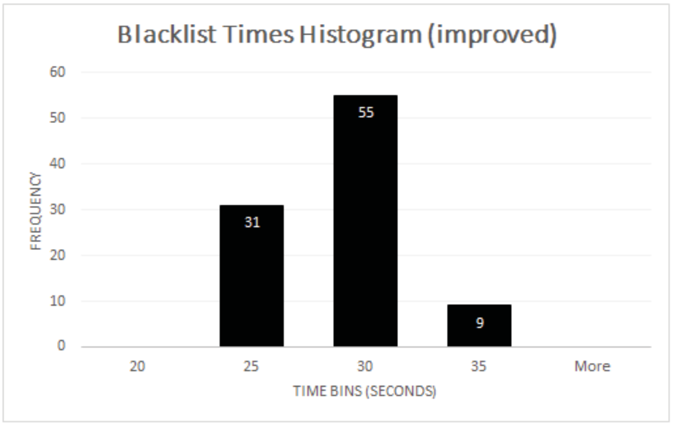

Fig. 3. Improved response time frequency distribution, 5-second bins.

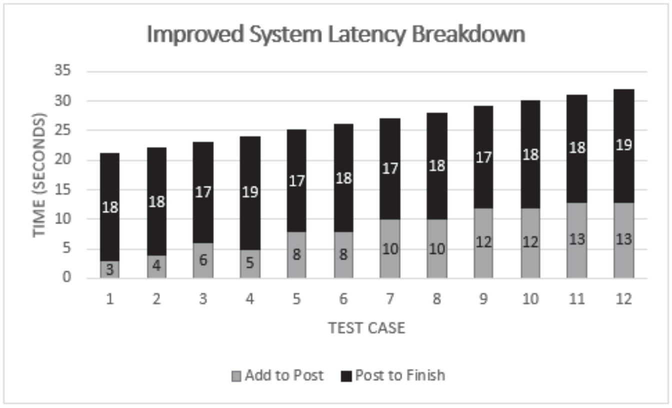

Fig. 4. Distribution random sample from 10-second bins, showing add-to-post and post-to-finish response times.

The revised latency distribution and timing breakdown (in 5-second bins) are shown in Fig. 3 and Fig. 4. This resulted in both lower total response time and a tighter response time distribution; we did not attempt further optimization in this proof-of-concept test bed.

In our testing, attempted DDoS attacks are blocked immediately after the first intrusion attempt by the BlackRidge gateway and firewall. By changing the configuration parameters for the authentication appliance, firewall, or both, we can adjust the length of time that a given set of IP addresses remain blocked. This also changes the point in the network data plane where these addresses are blocked. For example, we detected and blocked DDoS attacks for one minute with BlackRidge ACLs. Blackridge then began to accept traffic from suspect IP addresses. That minute is ample time for our system to reconfigure the downstream firewalls to permanently block traffic from suspect IP addresses.

Note that while we may allow suspect IP addresses to resume authentication attempts against the BlackRidge gateway, we can dynamically adjusting the trust level for these packets (reducing the trust level of the IP addresses associated with a prior DDoS attack). To control trust levels on the BlackRidge authentication gateway, we used the API built into version 3.1 of the management software. As shown in Fig. 1, we can also use the orientation step from Graylog and the orchestration server to trigger a call to the BlackRidge gateway API, and dynamically adjust trust level for any of the devices with managed identities configured on the gateways. This might be employed in the case of insider threats when we need to adjust the trust level for authorized users based on potentially malicious traffic patterns observed from their systems. The response times for this adjustment are similar to those measured for dynamic adjustment of the Juniper API, and as before are limited by the response times of the GrayLog export events.

This approach offers several advantages over a conventional network security system. A system based only on firewalls with manual ACL reconfiguration and log analysis would be unable to detect and respond to a DDoS attack quickly enough to prevent the attack. Conventional system response times can be as long as several minutes [1], [2], [3], [4], as compared with under 32 seconds for our approach. Further, we provide increased flexibility in creating attack response recipes by orchestrating multiple pieces of network equipment (authentication gateways, firewalls) from multiple vendors with control plane APIs (open source and commercial devices). For the DDoS attack use case, we block attacks almost immediately using a very fast feedback loop in the authentication appliance (typically tens to hundreds of milliseconds), then reinforce this defense with a second outer feedback loop to the firewalls (typically well under a minute). Note that we have not optimized the response times for this test bed, and our analysis suggests that the outer feedback loop response time can be significantly improved by optimizing the open source log parser. A fundamental principle for cybersecurity involves creating OODA loops with response times fast enough to outpace attacks [8]. While our current experiments cannot claim to block all types of DDoS attacks, we find that our response time of a few tens of seconds represents a significant improvement [1], [2], [3], [4]. Conventional systems cannot keep up with this approach, while data on our autonomic system suggests that the defensive control loop response can be lowered to at least 32 seconds or less, which is a significant improvement.

IV. Conclusion

In this paper, we have demonstrated that autonomic OODA control planes for cyber-defense systems are more efficient than conventional methods. Dynamic orchestration of authentication gateway trust levels, as well as both physical and virtual firewalls, has been achieved. Our system effectively detected and blocked DDoS attacks (over 100 unauthorized access attempts within 60 seconds). The attacks were detected and blacklisted within tens to hundreds of milliseconds by the authentication gateway, which also implements FPA and TAC. Dynamic ACL orchestration at the firewalls was demonstrated with 21-32 seconds latency. Analysis of the latency components suggests that further reductions could be achieved with modifications to the export log functions in GrayLog. Future work in this area may include dynamic optimization of other network equipment (for example, SD-WAN devices to dynamically adjust bandwidth in response to a DDoS attack).

References

[1] Cisco Institution, “Cisco 2017 annual cybersecurity report,” Cisco, Tech. Rep., 2017.

[2] Mikko Hypponen. and Tomi Tuominen., “F-Secure 2017 State of Cybersecurity report,” F-Secure, Tech. Rep., 2017.

[3] M. Compasti, R. Badonnel, O. Festor, R. He, and M. Kassi-Lahlou, “A software-defined security strategy for supporting autonomic security enforcement in distributed cloud,” in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2016, pp. 464–467.

[4] A. Saxena, M. Lacoste, T. Jarboui, U. Lücking, and B. Steinke, “A software framework for autonomic security in pervasive environments,” in *Proc. of the 3rd International Conference on Information Systems Security*, ser. ICISS’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 91–109. [Online]. Available: http://dl.acm.org/citation.cfm?id=1779274.1779286

[5] NIST Security, “Developing a framework to improve critical infrastructure, cybersecurity,” National Institute of Science and Technology, Tech. Rep., 2013.

[6] J. Chaffetz, M. Meadows, and W. H., “The OPM Data Breach: How the Government Jeopardized Our National Security For More Than a Generation,” Oversight and Government Reform, Tech. Rep., 2016.

[7] G. Stoneburner, A. Goguen, and A. Feringa, “Risk management guide for IT systems,” http://csrc.nist.gov/publications/PubsSPs.html800-30, Sept 2012, online, Accessed 7/15/2017.

[8] J.Boyd, “OODA Model Summary,” http://www.valuebasedmanagement.net/methods_boyd_ooda_loop.html, Jan 2016, online, Accessed 7/15/2017.

[9] C. Decusatis, P. Liengtiraphan, and A. Sager, “Advanced intrusion prevention for geographically dispersed higher education cloud networks,” in *Proc. IEEE/ACM International Conference on Remote Engineering Virtual Instrumentation*, Mar 2017.

[10] “Amazon Web Services Identity and Access Management,” https://aws.amazon.com/iam/, Apr 2016, online, Accessed 7/15/2017.

[11] C. Decusatis, P. Liengtiraphan, A. Sager, and M. Pinelli, “Implementing zero trust cloud networks with transport access control and first packet authentication,” in *Proc. IEEE International Conference on Smart Cloud*, Nov 2016.

[12] C. Decusatis, “Nsf securecloud, year 2: autonomic cybersecurity using ooda loops,” in *Proc. NSF Enterprise Computing Conference (ECC)*, June 2017.

[13] Juniper Networks, “Understanding intrusion prevention system for SRX series devices,” https://www.juniper.net/documentation/en_US/junos/topics/concept/security-ips-overview.html, online, Accessed 7/15/2017.

[14] “GrayLog open source log management system,” https://www.graylog.org/, online, Accessed 7/15/2017.