Nov 14th, 10:05 AM - 10:30 AM

NIDS in Airgapped LANs--Does it Matter?

Winston Messer
Dakota State University, winston.messer@trojans.dsu.edu

Abstract

This paper presents an assessment of the methods and benefits of adding network intrusion detection systems (NIDS) to certain high-security airgapped isolated local area networks. The proposed network architecture was empirically tested via a series of simulated network attacks on a virtualized network. The results show an improvement of double the chances of an analyst receiving a specific, appropriately-severe alert when NIDS is implemented alongside host-based measures when compared to host-based measures alone. Further, the inclusion of NIDS increased the likelihood of the analyst receiving a high-severity alert in response to the simulated attack attempt by four times when compared to host-based measures alone. Despite a tendency to think that networks without cross-boundary traffic do not require boundary defense measures, such measures can significantly improve the efficiency of incident response operations on such networks.

Disciplines

Information Security | Management Information Systems | Technology and Innovation

NIDS in Airgapped LANs–Does it Matter?

Winston Messer
Beacom College of Computer and Cyber Sciences
Dakota State University
Madison, SD, United States
winston.messer@trojans.dsu.edu
https://orcid.org/0000-0003-1684-7794

***Abstract*—This paper presents an assessment of the methods and benefits of adding network intrusion detection systems (NIDS) to certain high-security airgapped isolated local area networks. The proposed network architecture was empirically tested via a series of simulated network attacks on a virtualized network. The results show an improvement of double the chances of an analyst receiving a specific, appropriately-severe alert when NIDS is implemented alongside host-based measures when compared to host-based measures alone. Further, the inclusion of NIDS increased the likelihood of the analyst receiving a high-severity alert in response to the simulated attack attempt by four times when compared to host-based measures alone. Despite a tendency to think that networks without cross-boundary traffic do not require boundary defense measures, such measures can significantly improve the efficiency of incident response operations on such networks.**

***Index Terms*—intrusion detection, airgap, NIDS, Security Onion, Suricata, security regulation**

I. Introduction

Network-Based Intrusion Detection Systems (NIDS) are a commonly-deployed boundary-defense mechanism. They employ signature-based or anomaly-based detection methodologies [1]. Because of their position at the network boundary, and because they are privy to data-in-motion for analysis, indicators of compromise distributed by threat intelligence organizations and the output of threat intelligence tools often include rules for integration into NIDS engines [2].

However, not every network has a cross-boundary connection. Airgapped networks are networks with no external interconnects. They do not connect to the public Internet at all. They are chosen for use in certain high-security applications such as sensitive government networks and industrial control systems. Although airgaps provide benefits to security, they are not completely beyond the reach of cyberattacks. Numerous attacks have been demonstrated against airgapped networks [3].

Discussion in the scientific press of deploying NIDS in airgapped networks is limited to few papers and has so far been focused on SCADA systems in industrial applications, not to computer networks for typical office processing. Lopez Perez, Adamsky, Soua, and Engel [4] proposed the use of random forest-based machine learning to produce signatures to use in a NIDS in an airgapped SCADA network. Krause, Ernst, and Klaer [5] wrote recommendations in their analysis of SCADA networks in the critical energy sector to place NIDS within the network, especially at the network segment connecting the terminals used by human operators, rather than only at network boundaries. Maglaras, Jiang, and Cruz [6] proposed an anomaly detection-based intrusion detection system for SCADA systems and warn against an over-dependence on the airgap in ensuring the security of the network. Finally, González-Granadillo, González-Zarzosa, and Diaz [7] likewise warned against placing unearned trust in airgaps as the primary defense of SCADA systems.

Outside of industrial control, a main cause for establishing airgapped networks is to handle sensitive data. Such networks may be government or private, classified or unclassified, and may resemble a traditional office computing environment with the exception of an Internet connection [8]. Because such networks lack the specialization and unique equipment that SCADA systems require, solutions in the realm of industrial control may not suit their needs.

A. Network Design

This paper is intended to explore whether adjusting architecture to support the use of NIDS on airgapped networks that conduct office computing rather than industrial control is beneficial to their security posture. On peer-to-peer networks where only client systems are in use, configuring the network to allow the NIDS to sniff all packets may be the most reasonable solution. In networks with a division between client-facing and non-client-facing machines (such as file servers or web servers), this is the recommended position to place the NIDS. This subsection will show several possible technical and hardware configurations to enable this requirement to be met even on networks without a cross boundary connection. The remainder of this paper will experimentally show the possible benefits of this configuration on government networks.

In a network with no routing, the simplest means to enable NIDS on a small LAN is to allow packet sniffing on the network. This can be accomplished on a switched network with a monitor port and a network card in promiscuous mode on the NIDS system as shown in Fig. 1. If a physical or logical network hub is in use, simply placing the NIDS in promiscuous mode will suffice.

When virtualized systems are placed within a single virtual infrastructure, the most logical place to position the NIDS is within the virtual network with the virtual network configured

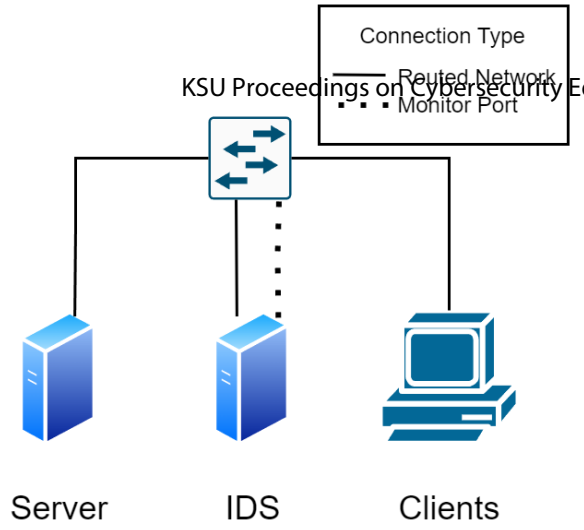

Fig. 1. IDS on a Monitor Port

to enable sniffing of packets traveling between the physical and virtual networks, as shown in Fig. 2.

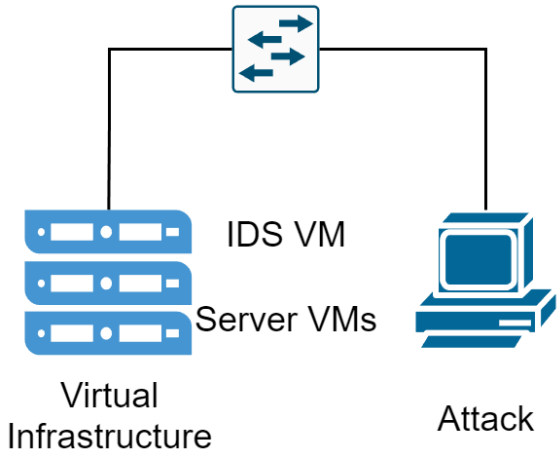

Fig. 2. IDS on Virtual Infrastructure

The rise of "next gen" firewall (NGFW) appliances that combine routing and NIDS functionality may enable a low-cost solution to the problem of integrating NIDS into a network. In such a case, the addition of routing can also enable the creation of VLAN separation and firewall rules between client-facing and non-client-facing systems. Fig. 3 depicts a configuration using an NGFW in place of a NIDS server.

If a router is already in place, a NIDS can be added placing the NIDS in the same VLAN(s) in which non-client-facing systems are deployed as shown in Fig. 4. Such a solution would be of similarly low cost to an NGFW-based solution as NIDS systems such as Security Onion can run on typical workstation or server hardware.

For the purposes of the experimentation conducted for this paper, a logical network most like Fig. 1 was used, with the entire network deployed as a virtual hub.

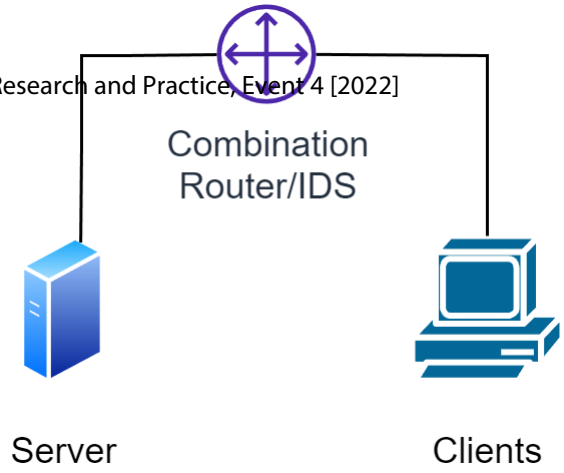

Fig. 3. IDS on Combination Router/IDS

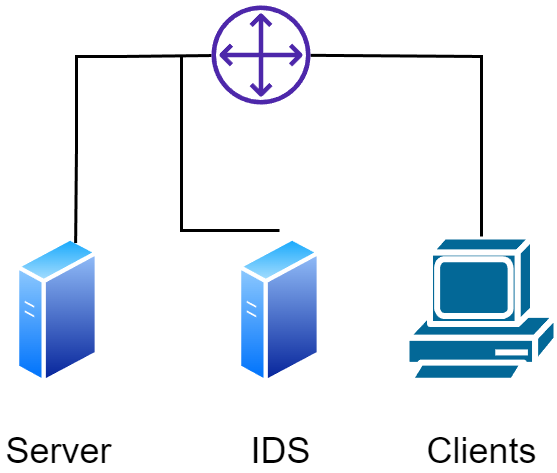

Fig. 4. IDS on Subset of VLANs In a Routed Network

II. Materials and Methods

A. Materials

A virtual network was created on commodity hardware to support experimentation. A "Monitor" system was created using Security Onion to provide network-based IDS functionality on the network. Security Onion is a Linux distribution that has been heavily customized to support intrusion detection and log monitoring as a network security platform [9]. It is designed with a wide spectrum of functionality and can be configured to provide NIDS support in airgapped networks. Security Onion provides the Suricata NIDS by default, and Suricata was the NIDS used for this effort.

OWASP provides a deliberately vulnerable virtual machine called "Broken Web Applications" or "BWA" for use in security education [10]. OWASP BWA was used to create the "Target" virtual machine on the network. BWA contains numerous vulnerable sites. "Bricks," a code-injection-focused site, was chosen for experimentation. Although airgapped networks do not have an Internet presence, malicious insiders

may pose a threat [11]. SQL injection attacks, such as the attacks to which OWASP BWA is vulnerable, have been considered as vectors for insider threats in the past [12]. Kali Linux was used as the "Attack" system.

1) Hardware: All experimentation was conducted with virtualization on commodity hardware. The system's processor was an Intel Core i7-2600 with a clock speed of 3.40GHz, and the system's memory was 32GB of DDR3 RAM. A 4TB Western Digital USB 3 external hard disk was procured to support the virtual machines and snapshots needed for this effort.

2) Software: The Host system was an x86_64 PC running Xubuntu 18.04LTS (amd64). VirtualBox was used for virtualization using version 5.2.42_Ubuntu r137960. The virtual machines used ran OWASP BWA version 1.2, Security Onion version 2.3.80, and Kali Linux version 2021.3. All software used for this effort was available at zero cost under free software licenses.

3) Experimental Testbed Design: The three above-named operating systems were installed in separate VMs per their respective instruction manuals. Security Onion was installed in "Airgap" mode to support the offline environment of the testbed. Wazuh was deployed to the Target system and connected to the Monitor system per the instructions provided by Security Onion for host-based IDS and log collection. The Kali system was installed from the .iso medium rather than from a pre-built virtual machine image. The three systems were established as follows:

- Attack System
  - Operating System: Kali Linux
  - Network Card 1: intnet803, IP address 10.8.3.4
  - Network Card 2: Network Address Translation to public Internet
    - Disabled during attacks
  - Network Card 3: Host-Only Adapter
    - Disabled except for during file transfers
- Target System
  - Operating System: OWASP BWA
  - Network Card 1: intnet803, IP address: 10.8.3.3 • Monitor System
- Monitor System
  - Operating System: Security Onion
  - Network Card 1: intnet803 (Management), IP address: 10.8.3.2
  - Network Card 2: intnet803 (Promiscuous Mode)

An internal-only network was established with the name "intnet803," and all three systems were connected to it. Promiscous mode access was granted to the Monitor system network-wide. The testbed network environment is depicted in Fig. 5.

4) Time Synchronization: To support the need to monitor events taking place across three separate virtual machines, a single synchronized time was required across the experimental network. Attempts to set the time once at install time and rely on software clocks did not prove reliable. Testing without an NTP server was likewise found to be unsustainable due to clock drift on the Target system and due to clock issues when restoring from suspension on the Monitor system.

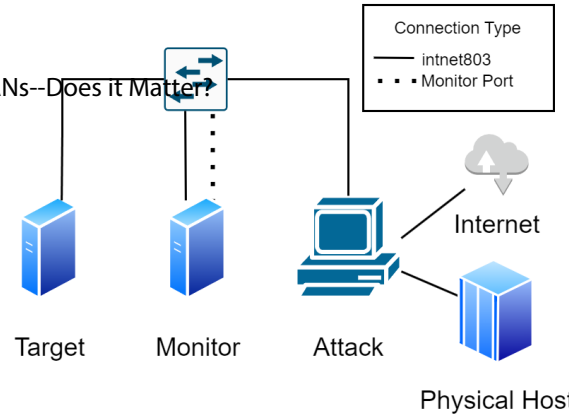

Fig. 5. Testbed Network

Once Kali Linux was installed to the Attack system, VirtualBox Guest Additions was installed to support quality of life improvements relating to system time, screen resolution, and shared clipboards. The Attack system did not experience clock drift relative to the Host system due to the installation of VirtualBox Guest Additions' enabling hardware clock synchronization. The Host system was itself synchronized to actual time via NTP. An NTP server was deployed to the Attack system, and the Monitor and Target systems were connected to it to synchronize time across all log files. While the Attack system was not the most realistic place to deploy NTP for the sake of simulation, its external connection and standardized operating system made it an ideal choice for the sake of simplicity. As NTP was not in-scope for attacks it was deemed not to pose a threat to the realism of the results.

B. Methods

1) Simulated Cyber Attack Setup and Execution: Due to the large attack surface area presented by OWASP BWA, the decision was made to limit the scope to OWASP Bricks, one of many security testing sites available within the BWA system. As an open source project, it is well-documented by official and unofficial sources. Bricks contains three classes of testing vulnerabilities: login bypass, malicious code upload, and code injection in site content.

Each testing target was tested in turn except for "Login 6" due to a known bug in this target. To increase repeatability and reduce bias in attack methodology, an existing third-party walk-through from pentester.land was followed to complete each challenge [13]. For the sake of realism, a single automated vulnerability reconnaissance scan was run prior to executing the walk-through.

OWASP ZAP is included by default in Kali Linux. It is a free and open source tool for testing web security. Although pentester.land leveraged the commercial tool Burp for its walk

through, ZAP was chosen for this effort due to its permissive license. The features used for this effort are available in both ZAP and Burp—as well as several other security testing tools—and the instructions can easily be adapted to the tester's tool of choice. The ZAP HUD-enabled Firefox browser provided by Kali was used for all aspects of this experiment. Prior to beginning the attacks, 10.8.3.3/owaspbricks/.* was added to the ZAP attack context, and the pages of OWASP Bricks were crawled.

2) Reconnaissance Vulnerability Scan: The "Active Scan" button was used to begin an automated scan for vulnerabilities of the Bricks site within the context permitted above. ZAP made 2,362 requests in 838 seconds.

3) Login Bypass Attacks: The walk-through was followed, with a thorough diary recounting of the attacks in Appendix B. Login 1 was directly attacked in the password field. Login 2 – Login 5 were attacked using ZAP breakpoints. Login 6 was skipped due to a known bug in this target that prevented an accurate test. Successful payloads are described in Table 1.

TABLE I
LOGIN BYPASS ATTACK

Attack	Attack Method	
	Payload Delivery	***Payload***
Login 1	In-Form Password Field	test' or 1=1 – -
Login 2	ZAP Breakpoint	username=three&passwd=four' or 1=1#&submit=Submit
Login 3	ZAP Breakpoint	username=one&passwd=two') or 1=1#&submit=Submit
Login 4	ZAP Breakpoint	username=one&passwd=test\\\") or 1=1#&submit=Submit
Login 5	ZAP Breakpoint	username=one' or 1=1#&passwd=two&submit=Submit
Login 6	No Test	N/A

4) Malicious Code Upload Attacks: OWASP Bricks' instructions suggest using an actual remote shell program rather than a proof-of-concept program. The b374k-shell was procured following Bricks' instructions and was substituted for the proof-of-concept programs used in pentester.land's walk-through. In all three Upload scenarios, b374k-shell was successfully uploaded and accessed. Required methods to upload this payload are described in Table 2.

TABLE II
MALICIOUS CODE UPLOAD ATTACKS

Attack	Attack Method	
	Payload Delivery	***Payload***
Upload 1	Direct Upload	[Unchanged]]
Upload 2	Direct Upload, MIME Type changed with ZAP Breakpoint	image/png
Upload 3	Direct Upload, MIME Type changed with ZAP Breakpoint	image/png

5) Code Injection in Site Content Attacks : The "Content" targets in OWASP Bricks ask the user to inject malicious code via site content outside of a login form or file upload page. The pentester.land walk-through was followed successfully in each Content attack. In each case, the intended data leakage from the Bricks website was obtained. The methods are documented in Table 3.

TABLE III
CODE INJECTION IN SITE CONTENT ATTACKS

Attack	Attack Method	
	Payload Delivery	***Payload***
Content 1	URL Bar	?id=0 and 1=2 union select load_file(0x2f6574632f706173737764),2,3,4,5,6,7,8 from mysql.user
Content 2	URL Bar	?user=harry' and 1=2 union select group_concat(name),group_concat(password),3,4,5,6,7,8 from users–-id=0 and 1=2 union select load_file(0x2f6574632f706173737764),2,3,4,5,6,7,8 from mysql.user
Content 3	URL Bar	username=harry' and 1=2 union select group_concat(name),group_concat(password),3,4,5,6,7,8 from users–-id=0 and 1=2 union select load_file(0x2f6574632f706173737764),2,3,4,5,6,7,8 from mysql.user&submit=Submit
Content 4	ZAP Breakpoint in User Agent	User-Agent: ' and 1=0 union select @@version,2,3,4,5,6,7,8– -
Content 5	Valid login using "tom" credentials retrieved in successful Content 2 attack above, followed by ZAP Breakpoint in Cookie	Cookie: User=tom' and 1=0 union select @@version,2,3,4,5,6,7,8– -;
Content 6	URL Bar	?id=MyBhbmQgMT0wIHVuaW9uIHNlbGVjdCBAQHZlcnNpb24sMiwzLDQsNSw2LDcsOC0tIC0="

III. RESULTS

For each case, an event was considered to have sufficient logging for incident response if log files provided the text of the malicious actions and a means to correlate the malicious activity with the IP address of the attacking machine and the time of the attack. This is because in the kinds of airgapped networks the experimental test bed is supposed to represent users do not bring their own devices. Therefore, an IP address and a timestamp should be sufficient to identify the hardware used and retrieve the login history for that hardware. For host-based detection, the Apache access and error log files and the MySQL log file provided this visibility into events.

An alert was considered to be raised in network-based detection if Security Onion displayed the log message on its "Alert" page. The only sensor in Security Onion to raise an alert in network-based detection was Suricata, the network intrusion detection system (NIDS) that Security Onion uses by default. For host-based detection, ModSecurity, an Apache module and web application firewall, provided the only alerts that were raised on the host during testing.

A. Reconnaissance Vulnerability Scan

1) Network-Based Detection: The scan caused Security Onion to display 2,297 high severity results and 302 medium-severity results for a total of 2,599 alerts as shown in Table 4.

TABLE IV
NETWORK-BASED DETECTION OF RECONNAISSANCE VULNERABILITY SCAN

Count	Alert Information	
	Severity	***Rule Name***
2,252	High	ET POLICY Http Client Body contains passwd= in cleartext
260	Medium	ET WEB_SERVER SQL Errors in HTTP 200 Response (error in your SQL syntax)
21	Medium	GPL WEB_SERVER 403 Forbidden
21	Medium	GPL WEB_SERVER .htaccess access
16	High	ET WEB_SERVER CRLF Injection - Newline Characters in URL
14	High	ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt
10	High	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
4	High	ET WEB_SERVER PHP Possible https Local File Inclusion Attempt
1	High	ET DELETED Redkit Java Exploit request to .class file

2) Host-Based Detection: The scan caused Mod Security to create 21 alerts.

B. Login Bypass Attacks

1) Network-Based Detection: In all five executed test cases, the Security Onion's NIDS Suricata created a true-positive alert for "ET POLICY Http Client Body contains passwd= in cleartext." This alert is caused by the site design of having a field named "passwd" that returns an unprotected password. As modern site design should not do this, an alert is raised. The text of the alert raised in Security Onion includes the text of the attack which was sufficient to meet the criteria attribution in incident response. However, Security Onion did not raise an alert that the usage was an actual attack, only that the password weakness was detected.

2) Host-Based Detection: No alerts were raised. Sufficient information to attribute the incident to a specific machine and time was reached in each case. Two non-critical errors were reported in the Apache error log.

C. Malicious Code Upload Attacks

1) Network-Based Detection: In the three Upload attacks, Suricata raised an alert of type "ET WEB_SERVER SQLi - SELECT and sysobject" in response to accessing the malicious code. The act of logging into the malicious site raised an alert of type "ET POLICY HTTP POST contains pass= in cleartext" and a second alert of type "ET INFO Suspicious HTTP POST Only Containing Pass - Possible Phishing."

2) Host-Based Detection: In the first case, the only detectable output in host-based response was in the upload and access of the b374k-shell's PHP web shell. The web shell's payload succeeded in its entirety without triggering other log events. In the subsequent two test cases, ModSecurity raised alerts due to the mismatched MIME type of the uploaded malicious code. Additionally, in the two final test cases, Apache raised errors corresponding to failed uploads before the MIME type was changed for success.

D. Code Injection in Site Content Attacks

1) Network-Based Detection: Content 1 and Content 2 were detected as SQL injection attacks as shown in Tables 5 and 6, respectively.

TABLE V
NETWORK-BASED DETECTION OF ATTACKS AGAINST CONTENT 1

Count	Alert Information	
	Severity	***Rule Name***
1	Medium	ET ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)
1	High	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
1	High	ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM
1	High	ET WEB_SERVER SQL Injection Local File Access Attempt Using LOAD_FILE
1	High	ET WEB_SERVER Possible MySQL SQLi User-Dump Attempt

TABLE VI
NETWORK-BASED DETECTION OF ATTACKS AGAINST CONTENT 2

Count	Alert Information	
	Severity	***Rule Name***
1	High	ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM
1	High	ET WEB_SERVER MYSQL SELECT CONCAT SQL Injection Attempt
1	High	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT

Suricata raised a true-positive alert of "ET POLICY Http Client Body contains passwd= in cleartext" for Content 5, however Suricata was here detecting the weakness in the site design not the specific attack. However, sufficient context was provided in the detection alert to discover and attribute the attack without turning to other sources. No network-based alerts were raised for Content 3, Content 4, or Content 6.

2) Host-Based Detection: No alerts were raised by host-based monitoring. The Apache access log showed the access to the pages viewed, and MySQL showed the SQL requests being run. However, the request for /etc/passwd in Content 1 and Content 2 remained obfuscated in the log files rather than indicating the file being accessed.

IV. ANALYSIS AND DISCUSSION

An analyst equipped with both the network-based detection methods and the host-based detection methods was twice as

likely to receive an accurate, actionable, specific alert (one that correctly identifies not only the flaw exploited but the attack taking place as well) as one that had only access to host-based alerting (40% vs 20% of test cases, p=0.04). Such an analyst was four times as likely to receive any high-severity alert that indicated a security risk (80% vs 20% of test cases, p¡.01)

Although some of the alerts were raised by site vulnerabilities rather than by attacks, the vulnerabilities were confirmed to exist on the site. It is worth pointing out that an analyst equipped with these alerts might have been able to prevent some of the attacks by raising an investigation of the alerting page.

Host-based evaluation of log files remained the gold standard for incident reconstruction, however, with attribution being possible for the analyst equipped with only host-based logs in 100% of cases. Although this paper recommends only adding network-based detection to an existing host-based infrastructure, were an analyst to use only network-based detection, event reconstruction was possible in 80% of cases. Although in this experiment, the 80% of attacks that were detected and alerted by network-based means were able to be attributed without pivoting to log file analysis, it remains the intent of the proposed network design that network-based detection be added to host-based logging, not used in place of it.

V. Related Work

A similar network design was proposed and documented by the BlackGirlsHack.org Certified Ethical Hacker study group for educational purposes [14]. Given that virtual networks based on these same operating systems have shown promise for both experimentation and education, it may be prudent for an organization to produce a standardized build of these systems in virtual machine format to ease deployment for future efforts. Because these virtual systems are available under a permissive open source software license they could be repackaged and redistributed freely for future experimental or training purposes.

VI. Conclusion and Future Work

A. Conclusion

This paper presented a proposed architecture to incorporate NIDS into airgapped LANs. An experiment was conducted to demonstrate the benefit of this addition. Against a controlled sample attack of OWASP BWA, an analyst would receive the benefits of a doubled to quadrupled chance of receiving an alert containing the attack in progress when compared with log file analysis alone. While incident response could be conducted in 80% of test cases with only the NIDS data, log file analysis remained the gold standard, allowing response in 100% of cases, showing that NIDS should be used for alerting but not as a replacement for retaining logs for later analysis. These findings support the proposed modification to airgapped LAN architecture to incorporate NIDS into their security posture.

B. Future Work

This methodology could be expanded to test other incident response tools and practices. Further NIDS could be tested with this methodology against other areas of OWASP BWA. Because OWASP Bricks focuses on web code-injection attacks, future testing should take non-web attacks into account.

References

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 16–24, 2013, doi: 10.1016/j.jnca.2012.09.004.

[2] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," Computers & Security, vol. 72, pp. 212–233, 2018, doi: 10.1016/j.cose.2017.09.001.

[3] Z. Zhou, W. Zhang, Z. Yang, and N. Yu, "Optical exfiltration of data via keyboard led status indicators to IP cameras," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1541–1550, 2019, doi: 10.1109/jiot.2018.2842116.

[4] R. Lopez Perez, F. Adamsky, R. Soua, and T. Engel, "Forget the myth of the Air Gap: Machine learning for reliable intrusion detection in SCADA systems," ICST Transactions on Security and Safety, vol. 6, no. 19, p. 159348, 2019, doi: 10.4108/eai.25-1-2019.159348.

[5] T. Krause, R. Ernst, B. Klaer, I. Hacker, and M. Henze, "Cybersecurity in power grids: Challenges and opportunities," Sensors, vol. 21, no. 18, p. 6225, 2021, doi: 10.3390/s21186225.

[6] L. A. Maglaras, J. Jiang, and T. Cruz, "Integrated OCSVM mechanism for intrusion detection in SCADA systems," Electronics Letters, vol. 50, no. 25, pp. 1935–1936, 2014, doi: 10.1049/el.2014.2897.

[7] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures," Sensors, vol. 21, no. 14, p. 4759, 2021, doi: 10.3390/s21144759.

[8] S. Wong and A. Woepse, "Software development challenges with air-gap isolation," Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, doi: 10.1145/3236024.3275526.

[9] K. L. Moore, T. J. Bihl, K. W. Bauer, and T. E. Dube, "Feature extraction and feature selection for classifying cyber traffic threats," The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, vol. 14, no. 3, pp. 217–231, 2016, doi: 10.1177/1548512916664032.

[10] S. Yeom, D. Shin, and D. Shin, "Scenario-based cyber attack·defense education system on virtual machines integrated by web technologies for protection of multimedia contents in a network," Multimedia Tools and Applications, vol. 80, no. 26-27, pp. 34085–34101, 2020, doi: 10.1007/s11042-019-08583-0.

[11] M. Guri, M. Monitz, and Y. Elovici, "Bridging the air gap between isolated networks and mobile phones in a practical cyber-attack," ACM Transactions on Intelligent Systems and Technology, vol. 8, no. 4, pp. 1–25, 2017, doi: 10.1145/2870641.

[12] E. Merlo, D. Letarte, and G. Antoniol, "Insider and outsider threat-sensitive SQL injection vulnerability analysis in PHP," 2006 13th Working Conference on Reverse Engineering, 2006, doi: 10.1109/wcre.2006.33.

[13] "OWASP Broken Web Apps - OWASP Bricks Challenge Walkthrough," 10-Jul-2018. [Online]. Available: https://pentester.land/challenge/2018/07/10/owasp-broken-web-apps-owasp-bricks-challenge-walkthrough.html. [Accessed: 28-Nov-2021].

[14] "VirtualBox Home Lab Setup Instructions," BlackGirlsHack.org, Jan-2021. [Online]. Available: https://blackgirlshack.org/wp-content/uploads/2021/01/BGHVirtualBoxLabSetUp.pdf. [Accessed: 05-Dec-2021].