Frankenstack: Real-time Cyberattack Detection and Feedback System for Technical Cyber Exercises

Mauno Pihelgas, Markus Kont

This paper has been accepted for publication at the 2021 IEEE CSR Workshop on Cyber Ranges and Security Training (CRST), and the final version of the paper is included in Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR).
DOI: 10.1109/CSR51186.2021.9527923
ISBN: 978-1-6654-0285-9

Frankenstack: Real-time Cyberattack Detection and Feedback System for Technical Cyber Exercises

Mauno Pihelgas
Technology Branch
NATO Cooperative Cyber Defence Centre of Excellence
Tallinn, Estonia
mauno.pihelgas@ccdcoe.org

Markus Kont
Research and Development
Stamus Networks
Tallinn, Estonia
markus@stamus-networks.com

***Abstract*—This paper describes a situation awareness framework, *Frankenstack*, that is the result of a multi-faceted endeavor to enhance the expertise of cybersecurity specialists by providing them with real-time feedback during cybersecurity exercises and verifying the performance and applicability of monitoring tools during those exercises. Frankenstack has been recently redeveloped to improve data collection and processing functions as well as cyberattack detection capability. This extensive R&D effort has combined various system and network security monitoring tools into a single cyberattack detection and exercise feedback framework.**

Although Frankenstack was specifically developed for the NATO CCD COE's Crossed Swords exercise, the architecture provides a clear point of reference for others who are building such monitoring frameworks. Thus, the paper contains many technical descriptions to reduce the gap between theoretical research and practitioners seeking advice on how to implement such complex systems.

***Index Terms*—automation, cyber exercises, cyber ranges, Frankenstack, monitoring, NATO Cyber Range, real-time feedback, security training, technical architecture**

I. Introduction

Cybersecurity exercises (CSXs) are key to enhancing cybersecurity operator readiness while also improving situation awareness (SA) in the cyber domain. Crossed Swords (XS) [1] is an annual interdisciplinary CSX directed at training participants for responsive cyber-kinetic operations. Although NATO nations have begun acknowledging the necessity of both defensive and offensive cyber capabilities, there are few exercises that tackle such a controversial subject. XS is organized by the NATO CCD COE in the NATO Cyber Range and utilizes a hybrid approach between cyber-physical and simulated infrastructure.

Cyber-exercise-specific SA systems are designed to improve SA during cyber exercises. While traditional SA systems are oriented toward cyber defenders, CSX-specific SA systems have been designed to provide situation awareness feedback not only to cyber defenders, but all participating teams alike.

CSXs introduce some unique requirements. The exercise environment must support the SA systems which measure the performance of participants. It is also important to assess the learning experience of participants to improve future iterations of the exercise.

978-1-7281-5684-2/20/$31.00 ©2021 IEEE

Stealth is an important factor when conducting cyber operations. During XS, the target Blue team (BT) hosts and networks are closely monitored for any indicators of compromise (IoC) which are then narrated to the training audience for learning purposes. Since 2016, a small team of cybersecurity monitoring specialists, the Yellow team (YT) in the exercise jargon, has been working to develop and improve the real-time CSX-specific framework known as Frankenstack that automates the entire cyberattack detection and feedback cycle for the training audience. The initial version of Frankenstack was described in our 2017 paper [2]. Since then, XS has been used as a platform to further develop, test, and validate the framework. The open-source solution is widely applicable to any other CSX where standard exercise technical infrastructure monitoring capability is available.

A. Problem statement

Some of the problems that inspired the creation of the initial version of Frankenstack were due to issues with the participants' learning experience. During the earlier iterations of XS, the YT feedback sessions regarding detected malicious activity were presented only at the end of each day. Due to limited time, these sessions were only able to summarize the primary observations from that day. This did not suffice as the technical training audience needed direct and faster feedback about their actions to pinpoint mistakes as they happened. This immediate feedback also needed to be adequately detailed so that the participants could understand how and why a particular attack was detected.

Additionally, most of the data analysis in YT was done manually by team members. This was slow and sometimes inconsistent in how attacks were followed up. Thus, another objective was to provide uniform feedback that would be clear and understandable for all participants.

Moreover, since the defending BT in XS is mostly just passively observing the situation, it often remained unclear to the participants whether any of their recent attacks would have affected the security posture of the adversary. While this reflects experience in the real world, it did not help participants' learning experiences during the CSX.

Furthermore, commercial SA systems are often too expensive to be acquired solely for cyber exercises due to license fees, hardware cost, and vendor-specific knowledge. The proprietary detection logic in commercial tools is often unavailable which again restricts YT's ability to understand and provide meaningful explanations of detected attacks.

Finally, the initial version of Frankenstack was too complex and involved several overlapping utilities (e.g., syslog-ng and rsyslog). While each tool had its purpose, the extra multiplicity rendered the data pipeline difficult to set up and maintain. The primary aim of the redevelopment was to reduce complexity and replace the use of many smaller utilities with tailor-made solutions that are described in this paper.

B. *Structure*

The remainder of this paper is organized as follows: section II provides some general background information about the XS exercise, section III presents an overview of related work, section IV describes the improvements and the technical architecture of the newly developed Frankenstack framework, section V discusses our efforts at providing relevant real-time feedback and appropriate visualizations for exercise participants, section VI briefly outlines the collaboration with industry partners, section VII defines future work, and section VIII concludes the paper.

II. Exercise background

Crossed Swords is an annual CSX that has been developed and organized by the NATO CCD COE since 2014. Although it started out as a primarily technical exercise, it has evolved into an interdisciplinary cyber exercise that involves technical, strategic, operational, and legal training aspects. It features a fictional scenario involving two notional countries, Berylia and Crimsonia. While most cyber exercises focus on training the defensive capability for Blue teams, XS reverses the role of the training audience, who now assumes the role of the Offensive Cyber Operations (OCO) team that is exercising a responsive cyber-kinetic scenario. The OCO team must work in close cooperation to discover an unknown network, complete a set of challenges, and collect evidence from the network for proper cyberattack attribution. Under such conditions, attribution, especially in the cyber domain, is increasingly hard to establish [3]. Another goal for the OCO team is to stay as stealthy as possible to avoid being detected by the monitoring stack which is described in this paper.

To develop and carry out the exercise, multiple teams are engaged: game network and infrastructure development (Green team – GT); game scenario development and execution control (White team – WT); defending team user simulation (BT); exercise monitoring and situation awareness (Yellow team – YT); and exercise training audience (OCO team). Note, that the *OCO team* was previously referred to as the Red team (RT), but since XS 2020 the terminology has been updated to better reflect current policies. Due to the reversed roles of the participants, the *defending* BT is actually the adversary that prompted the OCO team's responsive action.

Since offensive measures often take place in a cyber-physical space, the XS scenario portrays this by offering a variety of challenges that require a diverse set of skills and effective communication between members of the OCO team. In addition to physical devices such as programmable logic controllers (PLCs), IP cameras, radio devices, the exercise mimics realistic computer networks with a variety of different hosts (e.g., servers, workstations, and network devices) and operating systems. For instance, the networking subteam is responsible for attacking network services, protocols and routing; the client-side subteam targets human operators and attempts to gain foothold in the adversary's internal network segments; web experts attempt to compromise web services, applications and any associated databases; the digital forensics subteam performs data extraction and artefact collection; and kinetic forces provide support in operations that require a kinetic element such as physical surveillance, hardware extraction, forced entry, target capture, etc.

It is important to distinguish XS from capture-the-flag exercises. The participating subteams are not competing with one another, but rather serve as dedicated segments of a single military detachment. The exercise scenario is developed in a way that all subteams must coordinate their actions and share intelligence to achieve their objectives and advance in the exercise environment.

III. Related work

Research on exercises with an emphasis on offensive operations (such as XS) is almost non-existent. This is likely due to the high sensitivity of offensive cyber operations. However, there is an increasing amount of research based on other defensive CSXs and work that describes CSX-specific SA systems. Although not directly applicable in XS, the CSX-specific elements in those tools could still be considered relevant.

Känzig et al. [4] sought to detect command and control (C&C) channels in large networks without prior knowledge of the network characteristics. They leverage the notion that while benign traffic differs, malicious traffic bears similarities across networks. They trained a random forest classifier on a set of computationally efficient features designed for the detection of C&C traffic. They verified their approach using the NATO CCD COE's Locked Shields exercise datasets. Results revealed that the if the LS18 Swiss Blue team had used the system, they would have discovered 10 out 12 C&C servers in the first few hours of the exercise.

[5] Klein et al. compared two different machine learning techniques—the unsupervised autoencoder and the supervised gradient boosting machine—on a partially labelled cyberdefense exercise dataset. Both techniques were able to classify known intrusions as malicious while, surprisingly, also discovering 50 previously unknown attacks.

In [6], Arendt et al. presented CyberPetri, a redesign of the pre-existing Ocelot SA tool [7] which was used to provide real-time SA during the 2016 Cyber Defense Exercise and provide high-level feedback to network analysts based on exercise target systems' service availability reports. The authors note scaling to large datasets as a limitation. The exercise participants' feedback revealed that the tool was useful for the exercise White team for high-level decision making, but that technical specialists were more interested in improved exploratory capability for specific events and time windows.

A paper [8] from Henshel et al. proposed a performance assessment model of human cyberdefense teams and verified its applicability during the Cyber Shield 2015 exercise. While exercise data was captured during the game, most of the analysis was done after the event. For future iterations, the authors stress the need for real-time analysis of the collected data to adapt training and assessment methods already during the exercise. The ability to analyze the collected data was the primary limiting factor, as operators were not able to keep up with the huge amounts of incoming data.

Maennel focuses on measuring and improving learning effectiveness at cyber exercises [9]. This follows work that was described in the learning feedback section of the initial Frankenstack publication [2]. Furthermore, a recent paper [10] by Ernits et al. discusses how technical event logs and metrics from the exercise game system can be transformed to measure skills and learning outcomes.

Another publication [11] on team-learning assessment by Maennel et al. proposes an unobtrusive method based on mining textual information and metrics from situation reports submitted by teams during cyber exercises. Since these reports are regularly filed by Blue teams as a part of the exercise, this approach would enable gathering relevant information without disturbing the teams by conducting regular surveys and questionnaires throughout the exercise.

Chmelař describes the analysis of the XS 2020 exercise data using the MITRE ATT&CK knowledge base [12] to create reports of the OCO team progress [13]. Although the reports were created as a post-mortem analysis, the author proposes that they could theoretically provide in-game overview and visualizations during the XS exercise.

IV. Frankenstack

The Frankenstack SA framework features a near real-time feedback loop for the OCO team participants: any OCO action that is discovered on the game network and target hosts is analyzed in the automated data processing pipeline and if considered malicious is reported back to the feedback dashboard as an indicator of compromise. This all happens automatically without any human-interaction from the YT operators, allowing OCO members to immediately try again to improve their attack technique to attempt avoiding detection. Naturally, YT is still present to continuously improve the detection capability, reduce false positive alerts, and make sure the entire framework works as intended.

The framework also provides information about participants' progress to the exercise leadership allowing them to control the pace of the scenario more precisely. Figure 1 illustrates Frankenstack's technical architecture and various data flows in the exercise environment.

A. External data sources

Frankenstack requires multiple external data sources for its operation: full network traffic mirror, event logs, network configuration information, and the host asset database.

1) Network packet capture: Receiving a network traffic mirror is crucial for network security monitoring components. The mirrored network traffic was provided via a GRE (Generic Routing Encapsulation) tunnel from the virtual switches in the VMware NSX Data Center software-defined networking solution. Capturing traffic from virtual switches instead of edge routers meant that Frankenstack's IDS component (i.e., Suricata [14]) also had visibility in internal network segments, not just traffic that was traversing between network routers and perimeter firewalls. All traffic was captured and indexed using the Arkime full packet capture and analysis tool which was later used by the YT operators to extract new IoC samples directly from the indexed PCAPs to improve existing detection capability [15]. Note, that Arkime was re-branded from Moloch in November 2020.

2) Event logs: We collected event logs from the in-game (*gamenet*) systems wherever possible (e.g., Event Logs from Windows, Apache and nginx web server logs, and syslog from Linux). Windows Event logging was extended with additional rules for Microsoft Sysinternals Sysmon [16] from public GitHub sources [17], [18]. Instead of implementing AppArmor and SELinux for enhanced Linux auditing, we opted to use a small library called Snoopy Logger [19]. This was because configuring AppArmor or SELinux typically involves increasing the base level of security on the system. However, we did not want to interfere or impair any of the pre-configured vulnerabilities that were planted on the target systems.

Such host instrumentation is difficult to sustain in a standard defense-oriented CSX with BTs as the training audience. If the aim is to give BTs full control of their gamenet infrastructure, then it is difficult to ensure that BTs do not disable or reconfigure these tools. However, as the XS training audience is the OCO team, then YT could maintain supervisory control of all BT systems and ensure a constant stream of event log data.

3) Asset collection: Another critical piece of information is up-to-date knowledge about various network configurations and hosts in these networks. Since this is highly specific to the exercise and the underlying technical infrastructure, we developed a custom script that extracts this information from the Cyber Range provisioning API and the VMware vSphere API. While the provisioning API contains the information about how networks and hosts should be configured when they are deployed by Ansible [20], it lacks any knowledge of changes introduced after the initial deployment. Therefore, the list of all provisioned hosts and any static IP information can be easily collected from the provisioning API. However, if a host has been configured to obtain an IP address using DHCP, then the actual IP address must be retrieved from the machine during runtime. IPv6 link-local addresses are also assigned

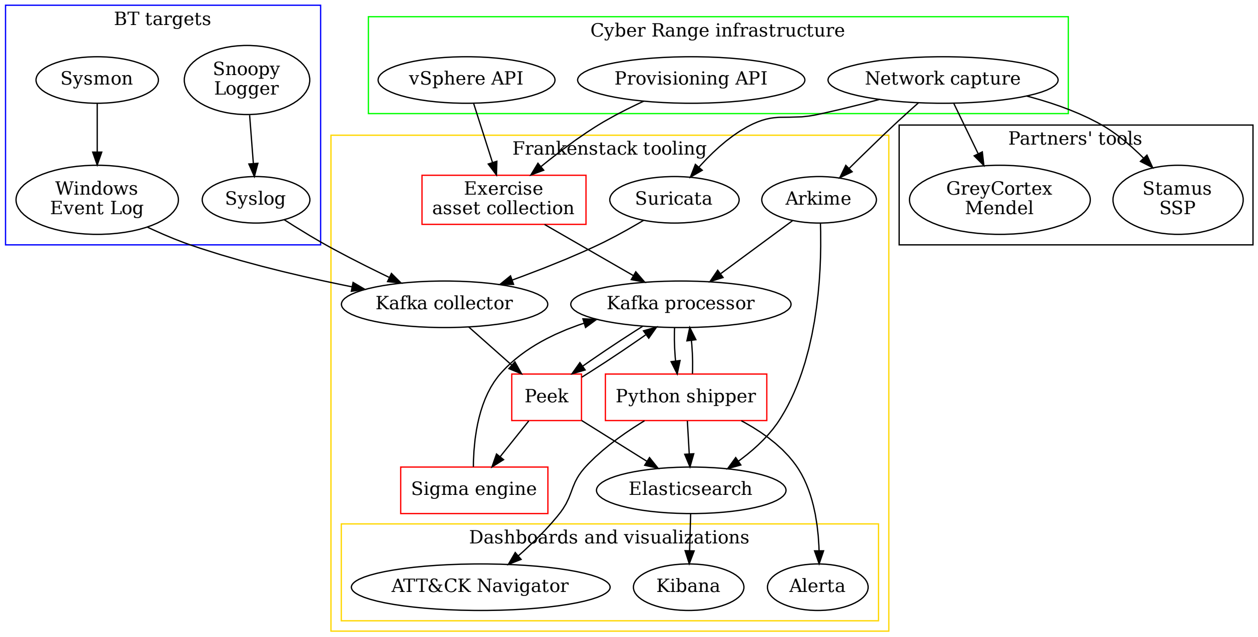

Fig. 1. XS technological perspective. Green area marks data sources originating from the exercise backend infrastructure, i.e., Green team assets (described in sections IV-A1 and IV-A3). Blue area indicates tools and data sources deployed on BT hosts and networks (described in section IV-A2). Black area displays partner tools (described in chapter VI). Yellow area illustrates Frankenstack components and the data flow between them (described in IV-B, IV-C and IV-D). YT Dashboards and visualizations are described in chapter V. Red nodes represent novel contributions developed solely by the authors of the paper.

dynamically. The script periodically collected the most recent IP information from all hosts using the vSphere API.

When information is gathered, we establish an asset profile for each known host in the exercise environment. Among other information, this profile maps all known IPv4 and IPv6 addresses to a particular host. This is key because when it comes to correlating events from multiple data sources, Frankenstack aims to be agnostic as to whether it has to match a hostname (e.g., from event logs) or on an IP address (e.g., from an IDS alert). This problem is something that many network monitoring solutions struggle with because it is not a trivial task to associate IPv4 and IPv6 network sessions even if they originate from the same host [21]. Trying to combine IDS alerts and event logs complicates things even further. For this reason, we attempt to enrich all events with an asset host name as a common identifier for all subsequent event correlation.

B. Distributed event streaming

The initial design of the event processing pipeline described in [2] relied primarily on Syslog-ng [22] to collect, store and forward events. This works well for systems and applications that are set up and configured beforehand so that proper syslog rules can be created. However, during the annual hackathons that precede the XS event, contributing partners in the YT would often need to integrate their own tools and scripts that needed to analyze the same incoming data feeds or a subset of past events to provide an alternative assessment to the main data processing pipeline. Having events stored as files on the central log collection server is not ideal for this. Alternatively, Elasticsearch [23] can be used to query historic data, but there is no good way to re-stream all incoming events in real time.

We analyzed various distributed message streaming tools (e.g., RabbitMQ and ZeroMQ) but opted for using Apache Kafka [24] as a central collection point for all emitted messages. Kafka fulfils the requirement for a multi-producer and multi-consumer event feeds. The general design concept in the Frankenstack framework is that all new events should be produced to Kafka and for any further processing, tools should consume the corresponding events from Kafka. Therefore, Kafka always has all the information and relevant stages of the data analysis processes.

C. Data processing components

Postmortem analysis of available datasets is an irreplaceable method during incident analysis. Although it often gives valuable insight about cyberattacks, it requires a substantial amount of time and manual work. Unfortunately, this conflicts with the short time span of the XS exercise and is not a viable method to keep track of the training audience. Frankenstack processes structured input data that has been sent to Kafka and applies event normalization, enrichment, and correlation for combining various information sources into a single stream of meaningful events.

All relevant event streams in Frankenstack have been configured to output structured JSON events at the source or to be transformed into JSON during the pre-processing phase. Unfortunately, the JSON events emitted by distinct sources feature varying structure which has to be individually handled for every input. Until 2019, Frankenstack employed SEC [25] as the primary data normalization, enrichment, and correlation tool. Bearing in mind that SEC and its rule language was initially designed for complex event correlation tasks on unstructured messages, it soon became difficult to handle complex nested JSON data structures in the SEC rule language. For example, any changes in input JSON key values resulted in the need to edit numerous textual rules. Therefore, instead of using the conventional SEC rule syntax, we had to write Perl code snippets into most FrankenSEC rules [26]. Although it was possible to accomplish what we required using custom Perl functions, the rule writing and management soon became infeasible to maintain.

Alternatively, processing complex nested data structures in a fully-fledged programming language seemed more approachable. In hindsight, our primary issue was that we attempted to correlate events too soon in the data processing pipeline—SEC is an event correlation tool, not a programming environment or a data normalizer. Unfortunately, in our ruleset we tried to accommodate many of the data processing and transformation tasks which should have been completed prior to sending events into SEC. This significantly hindered the SEC event correlation rule-writing process. The lack of proper post-processing meant that even minor changes in the input event structure resulted in the need to rewrite a large portion of the rules.

In the initial version of Frankenstack we had to integrate multiple logging tools (e.g., Syslog-ng, Rsyslog [27], and Logstash [28]) and custom scripts to implement a CSX-specific data normalization and enrichment tool. Configurations to process and route the event stream became overly complex as no single tool was readily able to satisfy all requirements. As a replacement we developed a novel data normalization and enrichment tool called Peek [29]. A fully customizable tool reduced this overhead significantly—we had to maintain only one tool which was fully under our control. Peek was able to replace generic log processing and event routing tools such as Logstash and Syslog-ng within our framework, albeit only for our particular exercise use-case.

The FrankenSEC ruleset was replaced by the Sigma ruleset [30] in XS 2020. Sigma is an open-source project that has gained a wide support and adoption from the information security community in the past few years. The project defines a simple rule structure in YAML format. Sigma does not do any pattern matching or alerting by itself. Rather, it acts as a technical translation layer and IoC sharing format. See Listing 1 for an example sigma rule that we developed for detecting base64 encoded scripts being executed in Windows machines. The rule triggers if the string *' -FromBase64String '* is detected within the *ScriptBlockText* field of a Windows Event.

Since Frankenstack aims to be vendor-agnostic, we built a custom real-time rule matching engine in Go [31] that uses Sigma rules. That engine is built as separate module and is publicly available in GitHub [32]. A detailed description and benchmarks of our Sigma rule engine are available in our recent whitepaper [33].

Listing 1. Sigma rule to detect base64 encoded PowerShell scripts.

```
title: Encoded ScriptBlock Command Invocation
author: Mauno Pihelgas
description: Detects suspicious PowerShell invocation
             command parameters
detection:
  condition: selection
  selection:
    winlog.event_data.ScriptBlockText:
    - ' -FromBase64String '
falsepositives:
  - Penetration tests
  - Very special PowerShell scripts
fields:
  - winlog.event_data.ScriptBlockText
id: 697e4279-4b0d-4b14-b233-9596bc1cacda
level: high
logsource:
  product: windows
  service: powershell
status: experimental
tags:
  - attack.execution
  - attack.defense-evasion
  - attack.t1059.001
```

The interface to the OCO feedback dashboards was accomplished with a comprehensive Python post-processing and event shipping module which ensured that all processed events sent to the dashboards (i.e., Alerta [34], ATT&CK Navigator [35] and Kibana [36]) conformed to a uniform structure and were mapped to the MITRE ATT&CK adversary tactics and techniques knowledge base. This post-processing script also implemented a simple baselining functionality to identify security events that occur under normal system use (e.g., execution of scheduled tasks, system updates, etc.). Later, during the exercise, such events were automatically filtered and not displayed on the dashboards. The same post-processing tool can be leveraged to easily create additional filters to suppress benign or false positive events which may occur during the exercise.

During the three days of XS exercise in December 2020, the Frankenstack framework received a total of 673,225 input security events. 85% of those were from Windows machines which are highly verbose, especially with the added Sysmon logging rulesets. The remaining 15% were logs emanating from Linux machines and Suricata IDS. To reiterate, manual inspection of such a large number of events for providing near real-time feedback to exercise participants would prove extremely difficult. Therefore, automated event processing steps such as filtering, correlation, and deduplication are of key importance within Frankenstack.

D. Determining the attacker

There is one crucial element that has to be determined for every event before submitting it further—the attacker and victim assets. To reiterate a problem from the early days of Frankenstack development, one primary concern the YT faced was automatically determining the direction of the attacks, i.e., identifying the victim and the attacker in a particular cyberattack. With Suricata IDS alerts, relying on the *source* and *target* fields does not yield an expected result—the *source* and *target* fields in IDS rules just signify the direction of traffic for which the detection match conditions are written. This meant that whenever the rule writers had written a rule that detects a response of an attack (e.g., sensitive data leaving the victim node), we would have erroneously classified the victim as the attacker. With this approach, we could only connect the relevant nodes, but lacked the directionality between them.

Fortunately, one of the core Suricata developers had been part of YT since 2016. He escalated this issue and for the following XS iteration there was a preliminary fix available. An improved solution has now been adopted into the mainstream version of the Emerging Threats (ET) rules [37]. ET rules now contain a metadata field called *attack_target*, which reveals the victim-side counterpart of the attack. Currently there are approximately 15,000 rules which contain the *attack_target* metadata keyword. This development has been presented at security conferences such as Hack.Lu [38] and SuriCon [39].

Peek now uses those keywords and enriches each atomic message with metadata to determine event shipper and, if applicable, proper event source and attack target. Our exercise asset collection tool enables us to build an asset database for threat hunting and map individual addresses from alerts to known assets. Source and destination information is inserted to every message metadata, along with event directionality flag (i.e., inbound, outbound, lateral, or local). The post-processing script is then able to process metadata-enriched messages and report affected assets to the feedback dashboards.

V. Visualizations

During XS, numerous large screens are installed in the training room directed at the OCO team. Their purpose is to provide visual feedback from various tools taking up any of the valuable screen real estate from the training audience.

Frankenstack comprises a set of open-source tools for visualizing log data, time-series metrics, and alerts. There are slight differences in handling various types of alerts: for example, alerts for CPU and memory usage trigger and recover automatically based on a predefined set of thresholds, while security events (e.g., IDS/IPS alerts) are only triggered based on some detection condition but lack the concept of a recovery threshold. Thus, such security alerts will never receive a recovery event, leading to an overabundance of information on the central dashboard. This requires special handling and conditions for timing out stale alerts.

Correlation and deduplication of recurring events is crucial for creating useful visualizations. Due to the volatile nature of CSXs, visualization tools can overflow with too much information for users to follow. For example, a network scan using the *nmap* tool can trigger a large amount of security events over a short period of time. While event correlation can collect and continuously combine those events, it should not wait indefinitely for the scanning to end before emitting the alert to the dashboard. The aim is to notify the OCO of their activity as it happens. Therefore, the length of the correlation window must be kept relatively short. With an effective deduplication functionality, sending the same alert multiple times does not cause any issues.

Alerta [34] is used as the primary feedback dashboard to present any malicious activity to exercise participants. The entire feedback cycle is completed by emitting the enriched event from the correlation engine to the Alerta dashboard. Each OCO team member has access to the Alerta API and web interface to create personal filtering rules for limiting the displayed information only to what is relevant in the current stage of the attack. We set a timeout to automatically archive stale events that had no correlated activity in the last 15 minutes to avoid flooding the dashboard with events.

Kibana [36] was used to present analytical dashboards that provided insight over the entire duration of the exercise, not just the recent events view available in Alerta. The information included a summary of detected attack types and statistics of IP addresses that have been generating the most alerts. The dashboards on the large TV screens were often observed by WT members who were more interested in the progress of the exercise and overall performance of the training audience.

We also mapped all feedback events to the MITRE ATT&CK framework attack phases and techniques. This enabled us to integrate the MITRE ATT&CK Navigator application to our environment and visualize the security events the OCO team had triggered to attack the target BT environment.

VI. Testbed for vendor appliances

Although Frankenstack has been kept open source, we have not denied cooperation with existing security platforms, SIEM systems, or commercial vendor appliances. Over the years several security vendors (e.g., Cymmetria [40], Greycortex [41], and Stamus Networks [42]) have joined the exercise YT to test their commercial products in a unique live-fire environment. We do not treat any proprietary security product as a component of Frankenstack, but rather as another monitoring data feed that can provide a different perspective into the exercise dataset.

VII. Future work

Several issues have remained a challenge. For example, to assess the progress of the OCO team more systematically, their attack campaigns would have to be incorporated into the exercise scenario itself. The exercise scenario and timeline would also have to be available in a machine-readable format so that Frankenstack could follow exercise progress and potentially adapt to non-technical scenario changes that take place as part of the storyline (e.g., dynamically adjust the level of detail provided within the feedback based on the participants' progress in the campaign).

More advanced visualizations are required for the exercise training audience and organizers to better follow the participants' progress at a higher level. The focus of Frankenstack has been on technical feedback rather than information that would provide actionable SA for higher-level decision makers. We attempted this with EVE [43] but faced the problem of revealing too much information too early in the game, which unfortunately limited its usability during the exercise.

VIII. Conclusion

The paper outlines the new developments of the cyberattack detection and feedback framework, *Frankenstack*. This open-source cyber-exercise-specific framework is based on a combination of various open-source monitoring tools. The primary purpose of Frankenstack is to provide detection of malicious activity and fully automated real-time observations during cyber exercises where the emphasis on training the offensive skillset.

The work describes the updated technical architecture compared to an earlier version of the framework. Furthermore, improved data processing, distributed event streaming, and feedback dashboards are described. Since 2017, we have implemented and verified the performance of our framework in the annual NATO CCD COE's Crossed Swords cyber exercise.

IX. Acknowledgements

The authors would like to thank Dr. Risto Vaarandi and Dr. Bernhards Blumbergs for their valuable advice.

This work has been supported by the Estonian IT Academy (StudyITin.ee).

References

[1] NATO CCD COE, "Crossed Swords Exercise," Available: https://ccdcoe.org/exercises/crossed-swords/, 2021.

[2] M. Kont, M. Pihelgas, K. Maennel, B. Blumbergs, and T. Lepik, "Frankenstack: Toward real-time Red Team feedback," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, October 2017, pp. 400–405.

[3] M. Pihelgas, *Mitigating Risks arising from False-Flag and No-Flag Cyber Attacks*. NATO CCD COE Publications, 2015.

[4] N. Känzig, R. Meier, L. Gambazzi, V. Lenders, and L. Vanbever, "Machine Learning-based Detection of C&C Channels with a Focus on the Locked Shields Cyber Defense Exercise," in *2019 11th International Conference on Cyber Conflict (CyCon)*, vol. 900, 2019, pp. 1–19.

[5] J. Klein, S. Bhulai, M. Hoogendoorn, R. Van Der Mei, and R. Hinfelaar, "Detecting Network Intrusion beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018, pp. 784–787.

[6] D. L. Arendt, D. Best, R. Burtner, and C. L. Paul, "CyberPetri at CDX 2016: Real-time network situation awareness," in *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, October 2016, pp. 1–4.

[7] D. L. Arendt, R. Burtner, D. M. Best, N. D. Bos, J. R. Gersh, C. D. Piatko, and C. L. Paul, "Ocelot: user-centered design of a decision support visualization for network quarantine," in *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, October 2015, pp. 1–8.

[8] D. S. Henshel, G. M. Deckard, B. Lufkin, N. Buchler, B. Hoffman, P. Rajivan, and S. Collman, "Predicting proficiency in cyber defense team exercises," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, November 2016, pp. 776–781.

[9] K. Maennel, R. Ottis, and O. Maennel, "Improving and Measuring Learning Effectiveness at Cyber Defense Exercises," in *Secure IT Systems: 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017. Proceedings*, November 2017, pp. 123–138.

[10] M. Ernits, K. Maennel, S. Mäses, T. Lepik, and O. Maennel, "From Simple Scoring Towards a Meaningful Interpretation of Learning in Cybersecurity Exercises," in *15th International Conference on Cyber Warfare and Security (ICCWS 2020)*, March 2020.

[11] K. Maennel, J. Kim, and S. Sütterlin, "From Text Mining to Evidence Team Learning in Cybersecurity Exercises," in *Companion Proceedings 10th International Conference on Learning Analytics and Knowledge (LAK20)*, March 2020.

[12] The MITRE Corporation, "MITRE ATT&CK," Available: https://attack.mitre.org/, 2021.

[13] M. Chmelař, "Utilizing MITRE ATT&CK to Create Adversary Reports of Live-Fire Cybersecurity Exercises for Feedback Purposes," Tallinn University of Technology, Tech. Rep., 2020.

[14] Open Information Security Foundation, "Suricata," Available: https://suricata-ids.org/, 2021.

[15] Arkime, "Arkime," Available: https://arkime.com/, 2021.

[16] Microsoft, "Windows Sysinternals - Sysmon," Available: https://technet.microsoft.com/en-us/sysinternals/sysmon, 2021.

[17] O. Hartong, "sysmon-modular," Available: https://github.com/olafhartong/sysmon-modular, 2021.

[18] SwiftOnSecurity, "sysmon-config," Available: https://github.com/SwiftOnSecurity/sysmon-config, 2021.

[19] B. S. Jese, "Snoopy Logger," Available: https://github.com/a2o/snoopy, 2021.

[20] Red Hat, Inc, "Ansible," Available: https://www.ansible.com/, 2021.

[21] B. Blumbergs, M. Pihelgas, M. Kont, O. Maennel, and R. Vaarandi, "Creating and Detecting IPv6 Transition Mechanism-Based Information Exfiltration Covert Channels," in *Secure IT Systems: 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings*. Springer International Publishing, 2016, pp. 85–100. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-47560-8_6

[22] One Identity LLC, "syslog-ng," Available: https://www.syslog-ng.com/, 2021.

[23] Elasticsearch B.V., "Elasticsearch," Available: https://www.elastic.co/elasticsearch, 2021.

[24] Apache Software Foundation, "Apache Kafka," Available: https://kafka.apache.org/, 2021.

[25] R. Vaarandi, B. Blumbergs, and E. Çalişkan, "Simple event correlator - Best practices for creating scalable configurations," in *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2015 IEEE International Conference on*, March 2015, pp. 96–100.

[26] NATO CCD COE, "frankenSEC," Available: https://github.com/ccdcoe/frankenSEC, 2019.

[27] Adiscon GmbH, "The rocket-fast Syslog Server," Available: https://www.rsyslog.com/, 2021.

[28] Elasticsearch B.V., "Logstash," Available: https://www.elastic.co/logstash, 2021.

[29] NATO CCD COE, "Peek," Available: https://github.com/ccdcoe/go-peek, 2020.

[30] F. Roth, "Sigma," Available: https://github.com/Neo23x0/sigma, 2021.

[31] "The Go Programming Language," Available: https://golang.org/, 2021.

[32] M. Kont, "Sigma rule engine," Available: https://github.com/markuskont/go-sigma-rule-engine, 2020.

[33] M. Kont and M. Pihelgas, *IDS for logs: Towards implementing a streaming Sigma rule engine*. NATO CCD COE Publications, 2020.

[34] N. Satterly, "Alerta," Available: http://alerta.io/, 2021.

[35] The MITRE Corporation, "Att&ck navigator," Available: https://github.com/mitre-attack/attack-navigator, 2021.

[36] Elasticsearch B.V., "Kibana," Available: https://www.elastic.co/kibana, 2021.

[37] Proofpoint, "Emerging Threats rules," Available: https://rules.emergingthreats.net/, 2020.

[38] E. Leblond, "Finding the Bad Guys, Yes Really," Available: https://www.youtube.com/watch?v=Scntdv1Vp_0, 2017, hack.lu 2017 Presentation.

[39] ——, "Finding the Bad Guys, Yes Really," SuriCon 2017, 2017, presentation.

[40] Cymmetria, "Cyber deception & NATO red teams," Available: https://cymmetria.com/white-paper/nato-crossed-swords/, 2018.

[41] Greycortex, "Greycortex supports Crossed Shields for second year," Available: https://www.greycortex.com/blog/greycortex-supports-crossed-shields-second-year, 2019.

[42] Stamus Networks, "Stamus Networks at XS20," Available: https://twitter.com/StamusN/status/1339968510924120066, 2021.

[43] F. J. R. Melón, T. Väisänen, and M. Pihelgas, "EVE and ADAM: Situation Awareness Tools for NATO CCDCOE Cyber Exercises," in *STO-MP-SCI-300 Cyber Physical Security of Defense Systems*, 2018, pp. 10–1–10–15. [Online]. Available: https://ccdcoe.org/uploads/2018/10/EVE-ADAM-MP-SCI-300-10.pdf