A Module for Anomaly Detection in ICS Networks

Matti Mantere
VTT Technical Research
Centre of Finland
Kaitovayla 1
90570 Oulu, Finland
matti.mantere@vtt.fi

Mirko Sailio
VTT Technical Research
Centre of Finland
Kaitovayla 1
90570 Oulu, Finland
mirko.sailio@vtt.fi

Sami Noponen
VTT Technical Research
Centre of Finland
Kaitovayla 1
90570 Oulu, Finland
sami.noponen@vtt.fi

ABSTRACT

Network security monitoring using machine learning algorithms is a topic that has been well researched and found to be difficult to use. We propose to use a specific approach in restricted IP network environments and leverage the network state information and information from individual connections for increased level of sensitivity. The approach is meant for use in restricted IP networks which exhibit a level of determinism that enables the use of machine learning approach. In this work we use algorithm called Self-Organizing Maps. We introduce an implementation of self-organizing maps engine built on top of the Bro network security monitor. An implemented selection of initial features for the Self-Organizing Maps is provided and a sample sub-set is used when training a SOM lattice for network data from an industrial control system environment. The anomaly detection prototype described in this paper is meant as a complementary mechanism, not a standalone solution for network security monitoring.

Categories and Subject Descriptors

H.0 [**Information Systems**]: General

Keywords

anomaly detection; cyber security; machine learning; network security monitoring

1. INTRODUCTION

In this paper we introduce a concept of using specific attributes of network state combined with individual connection information to provide meaningful features for network anomaly detection. The usage of network state attributes derived from a network security monitoring system as features for machine learning algorithm allows detection of changes in overall network environment, especially when combined with connection specific information.

HiCoNS'14, April 15–17, 2014, Berlin, Germany.

ACM 978-1-4503-2652-0/14/04 ...$15.00.
http://dx.doi.org/10.1145/2566468.2566478.

Anomaly detection in industrial control system (ICS) networks is an important issue given the increased connectivity of these environments and their high susceptibility to damage once their perimeter has been breached. They also present a tempting target for various attackers with motives ranging from mischief to industrial espionage and sabotage. Our contribution to this aims to improve the overall state of network security monitoring in industrial environments.

A deployable machine learning based anomaly detection system with passive monitoring would strengthen the existing security measures for a industrial control system network. It would require very limited manual customization before and during deployment. After deployment changes would only be needed when there are changes made to the network environment. Similar activities are ongoing by other parties, as evident in the Section 2, but our systems aims to leverage the existing rich network analysis framework to provide increased capabilities.

In addition to the introduction of usage of specific network state information we introduce an initial module implementation using Bro network security monitor (Bro, Bro NSM) [25]. Bro is used as a base system and source of the information for forming the normalized features of the machine learning algorithm part. Bro NSM is an open source network security monitoring and traffic analysis framework originally written by Vern Paxson and available for download freely [4].

This paper is limited to the description of these techniques, the implementation and early testing with clean data from the test network. The machine learning algorithm used in this approach is a modified version of the basic self-organizing maps (SOM). The SOM implementation is carried out using Bro's inbuilt scripting language. SOM uses unsupervised learning and is used to map multi-dimensional data into the two-dimensional map. The resulting mapping preserves the topological relation of the inputs and is also easily understood and visualized. [18]

Network state attributes can be used as a part of the feature vector of the SOM setup but are optional. The system depicted can also be used without these features or easily extended to accommodate new features as long as they are available for the Bro NSM system at the time of individual connection terminations.

The envisioned operational context for a system leveraging this approach is a restricted network such as ICS network or a similar environment. The applicability of machine learning based anomaly detection for these restricted environments is dependent on the individual network and the protocols found therein.

For the initial testing we used near ideal instance of such an restricted network environment. Specifically the packet capture files from the PrintoCent state-of-the-art printed intelligence pilot factory located at VTT Oulu premises, and more accurately the control network of a printing machine called MAXI [1]. This environment offers relatively deterministic data flows in daily cycles. The term *deterministic* is used in this paper in the similar manner as in the work presented in the paper [16].

Network environments with relatively deterministic or predictable properties also alleviate the problem with false positives, a common issue with anomaly detection systems [9]. The used target network and its relatively deterministic properties are explained in Section 3.5.

Based on our review of related work in Section 2 and the studies on systems available to public our system differs from most. The main difference is that compared to the more generic anomaly and intrusion detection solutions currently commonly available it has a very specific intended context of use that is coupled with automated modeling of the network traffic through machine learning algorithm and allows user to easily select the various parameters or even extend the functionality. It is not meant to be usable in more open network environments, which means that many of the issues faced by systems in those environments are avoided. The selection of features and functionalities are geared towards this specific purpose. It is also the first artificial neural network (ANN) module for Bro NSM that has been published, and benefits from the wide array of functionality and extendability inherent in the underlying system. The rich stateful nature of Bro also allows the use of network features otherwise difficult to obtain. We have not yet fully exploited the state information in our system.

This paper follows the standard structure. In Section 2 the most relevant related work is presented. In Section 3 the material and methods are described, including the early implementation of a test set of features, the general implementation of the system and a short description of the network from which the packet captures are received from. This is followed by the Section 4 on results and Section 5 for a discussion and eventually the Section 6 for the conclusion.

2. RELATED WORK

This section discussing the related work contains much of the same information as found in the papers published earlier on by the authors, lastly in the paper [23].

Significant work has been done on using machine learning algorithms for network security monitoring and anomaly detection. Self-organizing maps algorithm has been also been investigated for this particular purposes, such as in the work presented in [26]. Research presented in [27] investigates a multilevel hierarchical approach and comparison to a single level approach, which is what we use here. The work presented in paper [19] investigates the training of neural network anomaly detectors for intrusion detection purposes and also brings up the challenge of the variability present in most networks. Work in [17] investigates the usage of fuzzy SOM's in comparison the the classical implementation.

In the paper [28] Sommer *et al.* discuss the applicability of machine learning for network intrusion detection and difficulties arising from bringing the system out of the laboratory and into the open. This work has been instrumental for our work in mapping the difficulties of using machine learning for intrusion or anomaly detection. Concerning the aplicability of machine learning approaches to the monitoring and anomaly detection of industrial control system networks and other restricted networks we found several papers to describe work very important to ours. In work [16] Hadeli *et al.* investigate leveraging determinism in ICS network for anomaly detection, this stands in contrast to the challenge of diversity and variability for more open networks as stated in [19]. Linda *et al.* in paper [21] describe a neural network based method for detection intrusions in critical infrastructure settings with good test results. The authors of also investigate the issue in the light of leveraging machine learning for anomaly detection in ICS network context this work papers [24], [22] [23]. The paper by Yang *et al.* [32] also investigates the intrusion detection through anomaly -based approach presenting interesting work. The work presented by Lin *et al.* in [20] describes a new protocol parser for Bro specifically meant for a protocol used in supervisory control and data acquisition (SCADA) systems, namely DNP3. The advancement of the Bro in the field of SCADA systems is very important for the continuation of the work presented in this paper and therefore of great interest. The ability to parse most SCADA protocols would enable a machine learning approach to use protocol specific information as a basis of additional features. Important work on the Bro and work concerning leveraging the tool also includes work such as presented by the researchers in the following papers: [30][31][12][15]. Bro related information and downloads are available on the projects web site at [4].

The work presented in the paper [13] by Dreger *et al.* presents the idea of using host based context information for improving the accuracy of network intrusion detection systems (NIDS), and this might be something to be used as a source of features for the SOM implementation at a later stage.

Important previous work that we are building upon also includes other more generic seminal work done on intrusion and anomaly detection such as presented in the papers [10], [29]. A good treatise on more generic network security monitoring can be found in the book [8].

3. MATERIALS AND METHODS

The main software tool used in the work was the Network Security Monitor called Bro [25]. It is a very flexible and powerful system offering good extendability with its scripting language. A version 2.1 was used for the implementation of the SOM extension. The new module includes the SOM algorithm, needed functions for normalization and all the other functionality addressed to the handling of SOM and integration with Bro. The implementation is highly specific to Bro, due to the scripting language used which is intrinsic to Bro.

The scripting language was used to implement a instance of the SOM algorithm [18] with a number of features to select from at the end-users discretion. The feature set includes a number of connection specific features, as well as a number of features derived from the current state as reported by Bro internal functions. The highly stateful nature of the Bro allows us to use the network state information it has stored as additional features. As of this writing the implementation of the feature set is still work-in-progress, with only a limited number of features available. This will be further discussed in Section 3.1.

The system is taught to learn the normal connections and state of the network, and reports input which differs from traffic previously seen as an anomaly. The value which triggers this alert is user configurable variable which represent the minimum Euclidean distance between input and feature vectors that triggers an anomaly warning.

There are important Bro specific bits of information to be kept in mind. Such as that in addition to TCP traffic it also handles UDP and ICMP message flows as connections based on internal logic [25].This Bro behavior is reflected in this paper. When discussing ICMP or UDP connections, this Bro representation of the flows for these transport protocols is meant.

It is also important to note that the SOM implementation for anomaly detection is meant to be complementary to the usage of the network security monitoring and the Bro tool standard functionality, and not provide the capability to perform meaningfully effective intrusion detection by itself.

Other tools used for supporting the work include the applications used for packet capturing and analysis. Such as Tcpdump [7] and Wireshark[3].

The network traffic used for the proof-of-concept implementation was derived from the MAXI[1] and represents a novel setup with relatively pristine, deterministic and cyclic network environment. The network environment, it's specific design and the possibilities for network security monitoring therein are not in the scope of this paper and are only briefly described in Section 3.5.

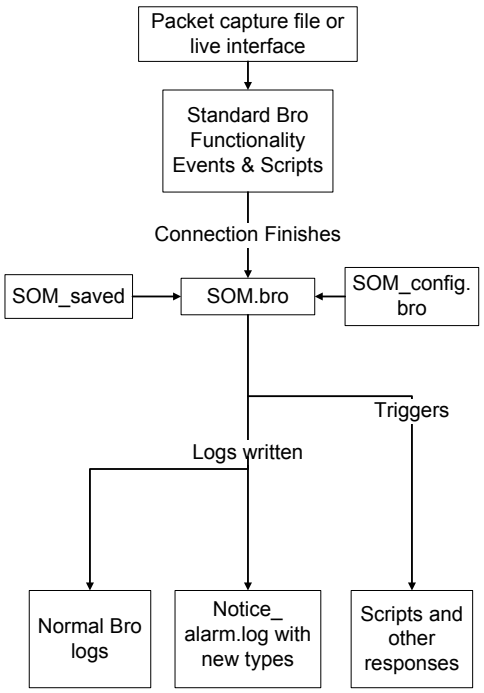

Figure 1: SOM extension for Bro NSM

3.1 Implementation

The current proof-of-concept level implementation as depicted in Figure 1 follows the standard SOM structure [18] with some tunable components. The implementation allows the envisioned end-user to tune the system to his or her needs by setting various attributes and variables defined in the configuration file.

The implementation is contained in two Bro script files, namely `SOM.bro` and `SOM_config.bro` with additional hook for calling the correct function in the connection handling script when a connection is terminated and to be logged. The built-in introspection function of Bro that produces information on the number of live connections is called right after the a connection terminates. The information from the introspection functionality is then combined with the information from the terminated connection to form the input vector.

Due to the limitations of the Bro scripting language the size of the SOM and the number of neurons that can be initialized is limited if done within Bro. Currently attempting to initialize overtly large lattices through recursive function calls results in a segmentation fault. In the future this initialization is to be done externally of the Bro by shell scripts or similar means.

The implementation as depicted in the Figure 1 consists of the core Bro functionality which has remained unchanged. The SOM functionality calls upon many of the Bro's internal functions for retrieving the network state at the time of the logging of a terminating connection. The terminating connection is processed both in the normal fashion of the standard system, as well as supplied to the SOM module for additional processing. The features selected for the use by the end user are then extracted. The features related to the terminating connection are processed and normalized, and the same is done on the selected network and Bro state features, which are obtained from the internal state of Bro by the provided functions. The initial features implemented are discussed in more detail in the Section 3.2. For the normalization, the largest single value seen since the creation of the specific SOM lattice is used. The seen value is divided by the largest value seen so far, which leads to all the values in input function receiving normalized values in the range of $[0, 1]$. This leads to initial instability as the initial normalization values tend to be far lower than the level at which they stabilize. This phenomenon fades out as the training is carried out in a restricted and deterministic network environment where the normalization values reach their approximate highest values after a single period or interval. However, it must be noted that currently there is no handling for possible outliers during the learning period, and this is something that will need to be looked at as it might cause sub-optimal normalization behavior.

The completion of a single connection acts as a trigger to call the SOM functions provided in the additional scripts. A hook for this activity is added to the `main` script of the `conn` module of Bro.

The features used can be selected by the user in the configuration file, currently `SOM_config`, which contains the feature vector that can be modified. The variables and structures that are meant to be modified by the user are also marked open to redefition in the Bro language. Therefore the user can also redefine the same variables again in other scripts as well, and not need to touch the original configuration file directly.

A detailed introduction of the SOM algorithm can be found in [18] and is only discussed in this paper to the ex-

Table 1: Currently implemented features for SOM extension

The number of live TCP connections at the moment of connection termination
The number of live UDP connections at the moment of connection termination
The number of live ICMP connections at the moment of connection termination
Duration of connection that terminated
Overall network fragments pending reassembly by Bro
The amount of data (bytes) sent by connection responder
The amount of data (bytes) sent by connection originator
Number of packets sent by the connection responder
Number of packets sent by the connection originator

tent that is required by the depiction of the approach and the implementation.

3.2 Features

The features extracted and combination of them used controls to a large extent the efficacy of a machine learning based anomaly detection system [21]. The decision on features is therefore one of the most interesting research topics. In the SOM module introduced in Figure 1 we propose to use Bro NSM internal state, interpreted network state as well as connection information as components of the feature vector used in the SOM implementation. The SOM_saved in the figure marking the saved SOM lattice.

Currently there is an initial implementation for the the features depicted in Table 1. The first three features the live tcp and udp connections and fragments pending reassembly during the termination of a connection triggering the SOM functions are based on the state information of the Bro NSM. The first four are network state features representing the Bro NSM's handling of UDP and ICMP flows as pseudo-connections [25]. The fragments that are pending reassembly reflects the Bro's internal state concerning the state of fragments it has seen.

The connection specific features implemented include the duration of an terminating connection, whether TCP, UDP or ICMP using Bro's definition of UDP and ICMP pseudo-connections as connections [25].

Features of data statistics of a connection, including packets and data sent to and fro between the connections originator and responder are also implemented. While the number of packets sent, the amount of data and durations each represent a single feature as depicted in Table 1 they are still normalized using the maximum corresponding value seen for the particular transport protocol in use for that connection. In the test network this resulted in significantly improved contrast between particularly UDP and TCP connections. In an earlier implementation the large maximum values for these features derived as seen in the TCP connections resulted in all UDP connections being normalized very close to zero and the contrast between different UDP connections for these features nearly vanished.

3.3 Learning phase

Prior to the use of the system for detecting anomalies a learning phase is required. Two modes are currently possible; manual and automatic. Saving of the modified SOM

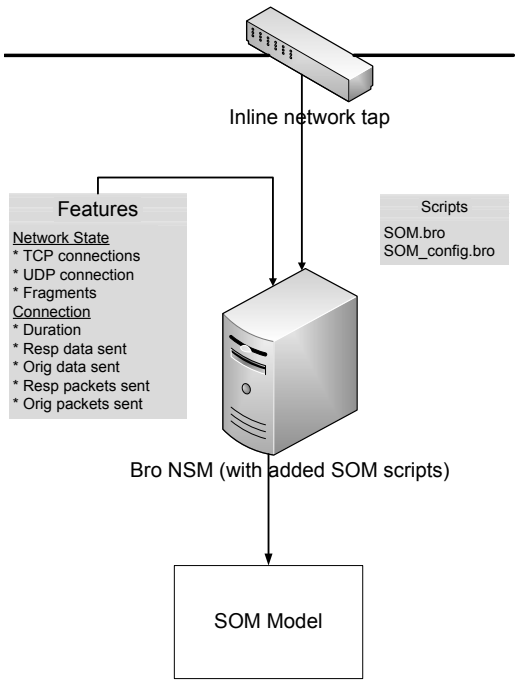

Figure 2: Depiction of SOM learning mode

lattice after learning phase is controlled by a flag, but running the system without saving the resulting SOM is mainly usable for debugging purposes.

In the automatic mode the system is set to switch to detection mode after a preset number of connections have been seen. The determination of the number of connections that is needed has to be decided by the user, and might result in a SOM lattice producing serious amount of false positives in the detection phase if not enough of connections are seen.

In the manual mode we repeatedly load the same saved SOM model and point to the packet capture file or network interface to be used for learning. Before each such learning run, the normal SOM variables can be tweaked, such as learning factor, the corresponding variables controlling the speed of the learning factor decrease, and the nature of the SOM neighbourhood function. As of this writing only one type of neighbourhood function is implemented, namely a 2-dimensional Moore neighbourhood of customizable size. The neighbourhood size is controlled by a variable that can be used to set the maximum Chebyshev distance for two dimensional plane with points having Cartesian coordinates: $D = \max(|x_2 - x_1|, |y_2 - y_1|))$ at which the neurons are still included in the neighbourhood. The 2-dimensional Moore neighbourhood used is also explained in the [18] for the basic SOM-implementation.

Figure 2 depics the learning phase of the system with some of the features already implemented. The inline network tap supplies connectivity to the network from which the traffic is to be learned by the SOM. In actual training situation, offline usage of a network traffic capture file would be more prudent.

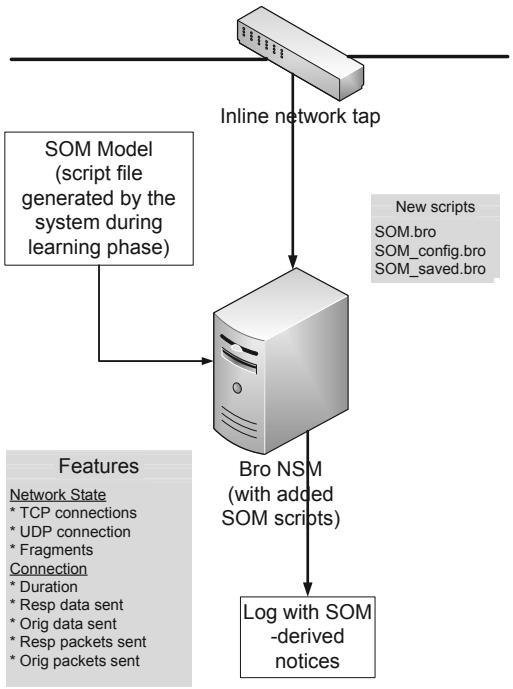

Figure 3: Depiction of SOM detection mode

Table 2: Notice report format for the SOM module

Possible_Anomaly_Seen Distance from nearest neuron[27,2]:[0.134853]

3.4 Detection phase

Figure 3 depicts the operational detection phase of the system. The detection mode requires that the learning mode has been run beforehand and the SOM model saved as a separate file, or the system is in automatic mode and the SOM map is in memory already. The SOM file also contains the features that were selected at the time of its initialization, and the SOM module retrieves this information from the file as well. For the detection several variables can be set to control the systems operation. The delta that marks the euclidean distance from the nearest SOM neuron that will trigger an alert should be set as low as possible to provide the desired balance of false negatives and false positives.

For the Bro logging framework additional alarm type has been defined for possible anomalies seen as deducted by the SOM algorithm. The new alarm type is accompanied by the additional information on which was the closest neuron in the SOM lattice for which the euclidean distance was still greater than `delta`. The exact euclidean distance from the nearest neuron is also logged in a format described Table 2.

The distance and the nearest neuron are logged for possible use to classify the connection, or deduce what type of traffic present in the network it most resembles. The code is to be instrumented in a way that will allow the user to flag certain clusters in the SOM as a type of traffic they represent. In the detection phase no changes are implemented in to the SOM model, this renders the system more resistant to slow attacks targeted towards the machine learning algorithm. During a slow attack the system is gradually taught to accept subtle changes as normal, but since the system is no longer learning anything new, this approach will not work.

When reacting to an attack or an anomaly raised by the SOM module, normal Bro NSM functionality is available, which provides a rich set of possible responses such as logging the incident or arbitrary triggering scripts.

3.5 Test network

Printocent is one of the world's first pilot factory environments for Intelligent Printing which is also capable of mass production. The latest printing line, called MAXI [1], was installed during 2012 and it can be used for prototype, ramp-up and small scale industrial production. Simplified network structure and the location of our monitoring tap is depicted in Figure 4. Simatics and Rexroth are controllers that read, and set new drive parameters to different units in the printing line. The parameters affect the quality of the end result of the printing process. Human-machine interfaces (HMI) are used to monitor and manually adjust parameters on the line.

Copper Tap -device from Black Box provides a complete copy of data from a full-duplex link to the network analyzer. The tap replicates the full-duplex signal from the network and sends one signal back to the network and the other signal to a monitoring laptop equipped with two distinct network interfaces. The Tap causes a 200-250 nanoseconds latency to the traffic passing through, but is otherwise completely transparent to the environment. There is no physical connection between the receive port on the analyzer side of the Tap and the Tap's internal processor.

Tcpdump [7] was used to record the traffic captures to libpcap packet capture format. Local traffic from the laptop is filtered out with tcpdump capture options and the capture interface is set to promiscuous mode. Two captures are automatically merged into daily capture files with scripts and mergecap which is a part of Wireshark [3].

The overall available network data consists of several months worth of traffic. The target network environment has several noteworthy attributes. The attributes are listed in Table 3. The start and shutdown are hardware controlled, specifically via one master switch controlling power. The network environment is relatively pristine, not having evolved much over time or since its inception. Due to the start in the morning and shutdown in the evening the system had a cycle length of around one short business day, from 6-8 hours on average.

Not all of the attributes listed in 3 are required for deployment of the system. The daily reboot cycle of start in the morning and shutdown in the evening is merely convenient, but by no means necessary. While it provides a clear daily cycle, the cycle could be of an arbitrary length, as long as the required traning data would contain the necessary amount of cycles for the target network. Having a relatively pristine environment is also more of a convenience for the development of our module, and not a requirement for deployment. However, having noise in the network can have an effect on the predictability of the network traffic.

The traces also consisted in large part of a proprietary Siemens protocols, namely S7 communication, for which there is no analyzer for in the Bro system. However, an open source plugin for Wireshark[3] is available at [5].

Table 3: Network attributes

Start in the morning, shudown in the evening
Environment is pristine
Low number of different protocols present in the network
Very stable and cyclic
Very deterministic or predictable throughout its daily cycles

These presence of these attributes make for a very controllable environment to test and developed network monitoring systems aimed at ICS networks and restricted network environments using internet protocl (IP) based communication schemes. Especially the deterministic nature is important [16].

The important downside of this is, that there are not that many pristine environments around, as most tend to be built over time and gone through multiple rounds of network evolution, as the systems have been replaced and added to. This typically leaves obsolete systems and protocols to use, which are required for compatibility, but have no other contribution for the overall system. This also contributes to the number of vulnerabilities found in these types of systems [11].

This issue creates a risk that the anomaly detection system developed for this type of an environment might require customization to be applicable to the more cluttered legacy systems. This might also add to the customization burden or even make the approach not feasible to use at all.

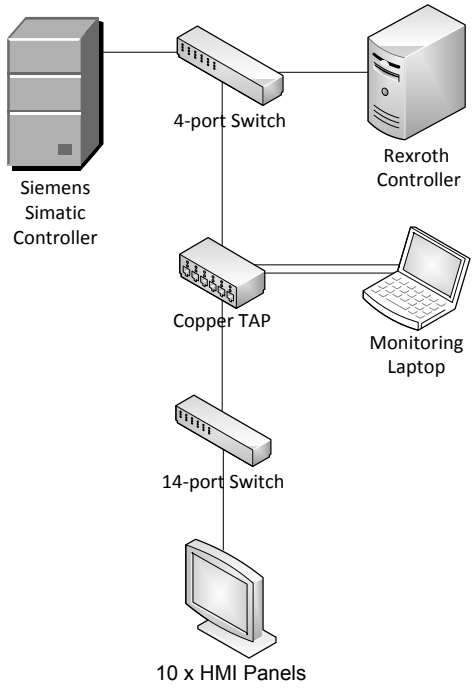

Figure 4: Simplified diagram of Printocent MAXI's network

For the initial testing of the algorithm we used several days worth of data, namely the packet captures ranging from 10. December 2012 to 31. January 2013 and some additional one day captures done during the time after this. Then we proceeded to test the system using the packet captures that had not been used during the training. The feature vector used in the initial testing is depicted in Table 4.

Table 4: Used feature vector

global features = vector("tcp_connections", "udp_connections", "duration", "orig_data_sent", "resp_data_sent") &redef;

The rate of false positives was mostly between 0-3 false alarms a day. These features did behave in a very deterministic manner, and the low number of false positives was not a surprise. However, whether these features can identify attacks and anomalies remains to be tested. That is to be the next phase of our work, and finding out what new features are needed and in what combination to catch as many of target anomalies and attacks as possible. The initial testing was also done with just a few test traces, as verification through testing was not in the focus of this paper.

The network packet rate that was seen in our monitoring location as depicted in the Figure 4 was an average of 68-73 packets per second during its uptime of generally 6-8 hours daily. The number of connections and pseudo-connections seen fluctuated, but typically around two to three thousand of them were seen during a typical day, including unnecessary traffic.

The reason behind this testing was merely to test that the system is implemented correctly, is capable of constructing the SOM lattice properly and provides anomalies when the euclidean distance threshold is violated. Elaborated results on testing the system with data including embedded attacks or attack-like activity are upcoming.

The further results with 6 clean and 5 dirty traces resulted in all the dirty traces yielding significantly increased number of alarms compared to the normal false positive rate. For example, an Nmap [6] operating systems detection traffic embedded in the test trace resulted in over 1000 new alarms being produced. This stands in stark contrast to the typical number of false positives reported in the initial tests. The promising test results are submitted for publication in a detailed form.

4. RESULTS

After the SOM was properly trained through several training runs with the capture data from the target network 4 several clusters formed as expected. The training took some manual tweaking of the various SOM learning factors, such as manipulation of neighborhood function size and the learning factor alpha, which controls the degree by which a winning weight vector in the SOM is moved towards the values of the input vector subject to learning. The learning mode worked as expected. A script wrapper to automatize the iterative process of teaching the SOM solves some of the issues, but the systems is susceptible to converge to a suboptimal stable state as is a possibility in SOM algorithm [14]. This behavior is difficult to predict, and requires at times the initialization of new lattices in a trial-and-error manner.

In the detection mode the system was configurable to yield a low amount of false positives, typically in the range from zero to three false positives a day when testing with a few later packet captures from the test network not used for training. The false positives typically were a result of one type of the long TCP connections terminating at the end of the day, for which only few occurrences were found in each days traffic. However, as this was just testing of the basic functionality, the actual operational false positive rate might be different.

In the learning mode the system was able to process a day-long capture from the MAXI network in roughly 30 seconds on a system with i5 Intel CPU, 8GB of memory and a SSD disk using a lattice of size $[x = 30, y = 20]$ and the same feature vector as depicted in Table 4. The Bro NSM currently utilizes only one core per running instance. Despite subtly differing computational tasks the detection mode took the roughly same amount of time when run on the same packet capture file.

5. DISCUSSION

Our previous work on anomaly and intrusion detection and machine learning in ICS context was carried out with different data. The data was received from a large factory installation which had gone through several changes during its time of existence. The nature of that traffic was very different to what we found when analyzing the MAXI data, and will be a target of further investigation if more traffic captures can be arranged to be procured from the site. The difference between a pristine network environment and one that has a considerable presence of obsolete components, systems and noise that has been built over time requires further study. This will be critical if the approach is to be of value to a greater number of end users.

It is also possible, that legacy networks exhibit characteristics that make it unfeasible to use this type of approach for them. The deterministic attributes are needed.

Also, it could be said that we are not actually using the network state attributes as features, but more the network state as interpreted by the Bro system. However, we feel that for simplicity's sake it is a close enough approximation, as Bro attempts to report the state as it sees it as accurately as possible.

The handling of outliers during the learning phase that might have an undesired effect on the normalization of input vectors must also be investigated further. Some heuristic to control the change of the normalization values will also need to be implemented. Further features for which a protocol analyzer is available in Bro are also to be investigated. Usage of the protocol parser investigated and presented in [20] is also to be undertaken.

It might also be prudent to divide the single SOM lattice into several lattices, each representing a single protocol or a set of protocol for which the features extracted are sufficiently similar for improved distribution after normalization. Currently the durations of the connections in the test network explained in Section4 are normalized with a value that leads to a very uneven distribution, with a bulk of the values very near to value 0, and the day-long TCP connections near value 1. This decreases the sensitivity of the system to slight changes in durations, since after the normalization the change is also very near to zero.

Inclusion of additional lattice to provide functionality for periodically checking the network state independent of terminating connections would improve the detection possibilities for on-going attacks.

6. CONCLUSIONS

We have implemented a proof-of-concept SOM engine using Bro NSM script and have tested the engine to work as expected. The performance evaluation needs to be commenced and more features implemented to improve the versatility of the system. The large feature set is meant for the user to select from to customize the system for his specific network. One set of features will not work for all types of networks.

The system is not designed to catch all attacks, but to complement existing solutions and the capabilities of network security monitoring solutions like the Bro NSM framework which it is built upon. The standard functionality of the Bro NSM system is available simultaneously with the SOM module, and the SOM modules information can be further processed by other scripts.

The SOM engine is currently operational, and efforts are ongoing to implement more features for accurate detection and the start of test runs for the MAXI[1] environment depicted in Figure 4. The exact sub-set of features to be selected from the total pool for the specific MAXI use case is also under investigations.

After further work and additional protocol specific features the system is expected to provide additional anomaly detection capability for existing ICS environments with minimal manual customization.

Acknowledgments

The research presented in this paper was mainly done as a collaborative effort in two research projects at VTT. Namely the DIAMONDS project [2], which is an EUREKA ITEA2 European Research Program project funded by TEKES, and INCYSE or Industrial Cyber Security Endeavour, which is a VTT internal research project.

7. REFERENCES

[1] PrintoCent, (accessed 1/6/2013). http://www.printocent.net.

[2] DIAMONDS ITEA2 Project, (accessed 17/3/2013). www.itea2-diamonds.org/.

[3] Wireshark, (accessed 2/5/2013). http://http://www.wireshark.org/.

[4] Bro Network Security Monitor, (accessed 2/6/2013). http://www.bro-ids.org/.

[5] S7comm wireshark dissector plugin, (accessed 6/6/2013).

[6] Nmap Network Security Scanner, (accessed 7/2/2013). http://www.nmap.org/.

[7] Tcpdump, (accessed 7/6/2013). http://www.tcpdump.org/.

[8] R. Bejtlich. *The Tao Of Network Security Monitoring: Beyond Intrusion Detection*. Addison-Wesley Professional, 2004.

[9] E. Cole. *Network Security Bible*. Bible (eBooks). Wiley, 2009.

[10] D. Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, SE-13(2):222 – 232, feb. 1987.

[11] DHS CSSP National Cyber Security Division. Common Cybersecurity Vulnerabilities in Industrial Control Systems, 2011.

[12] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer. Predicting the resource consumption of network intrusion detection systems. In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *Recent Advances in Intrusion Detection*, volume 5230 of *Lecture Notes in Computer Science*, pages 135–154. Springer Berlin / Heidelberg.

[13] H. Dreger, C. Kreibich, V. Paxson, and R. Sommer. Enhancing the accuracy of network-based intrusion detection with host-based context. In K. Julisch and C. Kruegel, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 3548 of *Lecture Notes in Computer Science*, pages 584–600. Springer Berlin / Heidelberg.

[14] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55, 1992.

[15] J. Gonzalez and V. Paxson. Enhancing network intrusion detection with integrated sampling and filtering. In D. Zamboni and C. Kruegel, editors, *Recent Advances in Intrusion Detection*, volume 4219 of *Lecture Notes in Computer Science*, pages 272–289. Springer Berlin / Heidelberg.

[16] H. Hadeli, R. Schierholz, M. Braendle, and C. Tuduce. Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration. In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8, 2009.

[17] W. Hu, D. Xie, T. Tan, and S. Maybank. Learning activity patterns using fuzzy self-organizing neural network. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(3):1618–1626, 2004.

[18] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[19] S. Lee and D. Heinbuch. Training a neural-network based intrusion detector to recognize novel attacks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(4):294–299, 2001.

[20] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer. Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 5:1–5:4, New York, NY, USA, 2013. ACM.

[21] O. Linda, T. Vollmer, and M. Manic. Neural network based intrusion detection system for critical infrastructures. In *Proceedings of the 2009 international joint conference on Neural Networks*, IJCNN'09, pages 102–109, Piscataway, NJ, USA, 2009. IEEE Press.

[22] M. Mantere, M. Sailio, and S. Noponen. Feature selection for machine learning based anomaly detection in industrial control system networks. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 771–774, 2012.

[23] M. Mantere, M. Sailio, and S. Noponen. Network traffic features for anomaly detection in specific industrial control system network. *Future Internet*, 5(4):460–473, 2013.

[24] M. Mantere, I. Uusitalo, M. Sailio, and S. Noponen. Challenges of machine learning based monitoring for industrial control system networks. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, march 2012.

[25] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435 – 2463, 1999.

[26] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection, LNCS*, pages 36–54. Springer Verlag, 2003.

[27] S. Sarasamma, Q. Zhu, and J. Huff. Hierarchical kohonenen net for anomaly detection in network security. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(2):302–312, 2005.

[28] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305 –316, may 2010.

[29] M. Thottan and C. Ji. Anomaly detection in ip networks. *Signal Processing, IEEE Transactions on*, 51(8):2191–2204, 2003.

[30] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney. The nids cluster: Scalable, stateful network intrusion detection on commodity hardware. In C. Kruegel, R. Lippmann, and A. Clark, editors, *Recent Advances in Intrusion Detection*, volume 4637 of *Lecture Notes in Computer Science*, pages 107–126. Springer Berlin / Heidelberg.

[31] N. Weaver, V. Paxson, and R. Sommer. Work in progress: Bro-lan pervasive network inspection and control for lan traffic. In *Securecomm and Workshops, 2006*, pages 1 –2, 28 2006-sept. 1 2006.

[32] D. Yang, A. Usynin, and J. Hines. Anomaly-based intrusion detection for scada systems. In *Proceedings of the 5th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies*, NPIC&HMIT 05, 2006.