**ABSTRACT**

The growing network connectivity witnessed in Supervisory Control and Data Acquisition (SCADA) systems raises cyber security concerns for Industrial Control System (ICS) facilities. To sustain critical infrastructure objective principles such as confidentiality, integrity, and availability from security breaches or devastating cyberattacks, compelling, proactive, and continuous security monitoring is needed. In this study, we propose a process to build an intelligent backend and visual system to handle real time data analytics. For that we demonstrate the use of the Security Information and Event Management (SIEM) tool, Splunk, to aggregate operational intelligence including network, system, and user behavior data. Also, to transform collected raw data into Indicators of Compromise (IOC) added intelligence data, we demonstrate the use of open source threat intelligence platforms. Real time analytics is then applied to prepared intelligence test data using MATLAB. With the proof of concept tool, Tableau, we present ICS system visual solutions, which can support security personnel to make decisions, understand concepts, or foresee the network problems.

ACKNOWLEDGMENTS

I would like to express my gratitude to the faculty of the Department of Mathematical, Computing, and Informational Sciences for their instruction during my program. It was an enjoyable and valuable learning experience.

I would also like to thank my major professor, Dr. Guillermo Francia, for his guidance and expertise, and also committee members Dr. Srinivasarao Krishnaprasad and Dr. David Thornton for their numerous contributions to this project.

I give special thanks to my family for their patience, support, and encouragement.

Suryapradeep Kavati

# Table of Contents

[Table of Contents iii](#_Toc499734197)

[List of Figures v](#_Toc499734198)

[List of Tables v](#_Toc499734199)

[1.0 Overview: 1](#_Toc499734200)

[1.1. Introduction: 1](#_Toc499734201)

[1.2 Purpose, Scope, and Objectives: 1](#_Toc499734202)

[2. Literature review: 2](#_Toc499734203)

[2.1 Machine Learning: 2](#_Toc499734204)

[2.2 Machine Learning model notation: 2](#_Toc499734205)

[2.3 Gradient descent algorithm: 3](#_Toc499734206)

[2.4 Linear regression algorithm 3](#_Toc499734207)

[2.4.1 For one feature variable 3](#_Toc499734208)

[2.4.2 For multiple features 4](#_Toc499734209)

[2.5 Logistic regression algorithm 4](#_Toc499734210)

[2.6 Neural networks 4](#_Toc499734211)

[2.6.1 Forward propagation 5](#_Toc499734212)

[2.6.2 Backpropagation 5](#_Toc499734213)

[2.6.3 The best approach to training neural network 5](#_Toc499734214)

[2.7 Support Vector Machine (SVM) 6](#_Toc499734215)

[2.7.1 SVM with kernels (Gaussian Kernel) 6](#_Toc499734216)

[2.8 K-means clustering algorithm 6](#_Toc499734217)

[2.9 Anomaly detection algorithm 7](#_Toc499734218)

[2.9.1 Anomaly detection vs. Supervised learning 7](#_Toc499734219)

[3. Threat Intelligence 8](#_Toc499734220)

[3.1 Intelligence lifecycle 8](#_Toc499734221)

[3.2 Types of Threat Intelligence 9](#_Toc499734222)

[3.2.1 Strategic Threat Intelligence 9](#_Toc499734223)

[3.2.2 Operational Threat Intelligence 9](#_Toc499734224)

[3.2.3 Tactical Threat Intelligence 9](#_Toc499734225)

[3.2.4 Technical Threat Intelligence 10](#_Toc499734226)

[3.3 Threat Intelligence Maturity Model: 10](#_Toc499734227)

[3.4 Threat Intelligence Platform (TIP): 11](#_Toc499734228)

[3.5 Cyber Kill Chain in ICS: 11](#_Toc499734229)

[3.6 Case study – Cyber kill chain process in the Ukraine Power Grid attack 14](#_Toc499734230)

[3.7 Cyber Kill Chain with Threat Intelligence Platform: 16](#_Toc499734231)

[3.8 Threat Indicators: 16](#_Toc499734232)

[3.9 Indicators of Compromise (IOC): 17](#_Toc499734233)

[3.9.1 IOC types 17](#_Toc499734234)

[3.9.2 IOCs in SCADA with CIMPLICITY HMI case study 18](#_Toc499734235)

[3.10 MineMeld Threat Sharing Tool 19](#_Toc499734236)

[4. ICS System Analysis & Design and Implementation 20](#_Toc499734237)

[4.1 Data Collection and Monitoring 21](#_Toc499734238)

[4.1.1 Data Collection Elements 21](#_Toc499734239)

[4.1.2 Data Collection and Aggregation: 22](#_Toc499734240)

[4.1.3 Threat Indicators collection from Alien Vault 23](#_Toc499734241)

[4.1.4 Data Normalization 25](#_Toc499734242)

[4.1.5 Data Synthetization with Abnormal Samples 25](#_Toc499734243)

[4.1.6 Enhancing Data Entropy 25](#_Toc499734244)

[4.1.7 Test Data Set 25](#_Toc499734245)

[4.2 Machine Learning Process 25](#_Toc499734246)

[4.2.1 Classification 25](#_Toc499734247)

[4.2.2 Evaluation of Outcomes and Observations 27](#_Toc499734248)

[4.3 Visual System Analytics 29](#_Toc499734249)

[4.3.1 Monitoring Available Space Checks in Disk Drive Type 29](#_Toc499734250)

[4.3.2 Monitoring More Frequent Instances of Processes on Host Machine 30](#_Toc499734251)

[4.3.3 Classifying Generated Events by Type of User 30](#_Toc499734252)

[4.3.4 Monitoring Event Logs Created by Users 31](#_Toc499734253)

[4.3.5 Analyzing Outbound Traffic of Client IPs 31](#_Toc499734254)

[4.3.6 Monitoring Destination IP Locations Geographically 32](#_Toc499734255)

[4.3.7 Observation of Bandwidth Usage 33](#_Toc499734256)

[4.3.8 Observing Windows Performance Monitor 33](#_Toc499734257)

[5. Conclusion and Future Enhancements 34](#_Toc499734258)

[6. References 35](#_Toc499734259)

[Appendix A – Data Collector Indexes for Splunk Input.conf file 37](#_Toc499734260)

[Appendix B – Test Data Features 40](#_Toc499734261)

[Appendix C – Python Script for IOC Collection 43](#_Toc499734262)

[Appendix D – Sample of Normalized Test Dataset 48](#_Toc499734263)

[Appendix E – Scripts for Data Classification 49](#_Toc499734264)

# List of Figures

[Figure 1: ICS Cyber Kill Chain Stage 1 12](#_Toc498275548)

[Figure 2: ICS Cyber Kill Chain Stage 2 13](#_Toc498275549)

[Figure 3: Attacker Objectives 14](#_Toc498275550)

[Figure 4: MineMeld working functionality 20](#_Toc498275551)

[Figure 5: Data collection, monitoring, and visualization system 23](#_Toc498275552)

[Figure 6:Algorithms 1 to 4 Confusion Matrix 26](#_Toc498275553)

[Figure 7: Algorithm 5 neural network confusion matrix 27](#_Toc498275554)

[Figure 8: Performance Comparison of Classifiers 28](#_Toc498275555)

[Figure 9: Monitoring available space checks in Disk Drive Type 29](#_Toc498275556)

[Figure 10: Distinct processes and instances of processes on host 30](#_Toc498275557)

[Figure 11: User created event code separation 30](#_Toc498275558)

[Figure 12: Visualizing number of logins/logouts 31](#_Toc498275559)

[Figure 13: Client IPs Outbound traffic 32](#_Toc498275560)

[Figure 14: Geographic locations of destinations IP 32](#_Toc498275561)

[Figure 15: Network bandwidth usage 33](#_Toc498275562)

[Figure 16: Network Transfers in Bytes 33](#_Toc498275563)

# List of Tables

[Table 1:Categories of Behavior Analytics 20](#_Toc498275582)

[Table 2: ICS/Scada System IOCs 24](#_Toc498275583)

[Table 3: Data Classification Results 28](#_Toc498275584)

# 

# **1.0 Overview:**

## ***1.1. Introduction:***

Presently, the internet is treated as a knowledge provider to our brain which helps people to make connections and to access information remotely. The popular Internet of Things utility and greater availability of electronic gadgets to public, people tended to use smart devices. So, the industries started to improve infrastructure from state to nation, nation to global by bridging industrial control system with information technology to provide smooth service to customers and reduce manned work. In general, the internet was built for connectivity, not for security means also allows intruders or cybercriminals to steal valuable information. Therefore, securing industry networks or maintaining security posture of an industry is a big challenge. The paradigm “Threat Intelligence and Machine Learning” approach might be helpful to protect Industrial Control Systems Security. Machine Learning approach involves training network traffic, generating a predictive model for anomaly detection, and identifying and visualizing present and forthcoming attacks with the help of alert system and threat intelligence.

## ***1.2 Purpose, Scope, and Objectives:***

Industrial Control System (ICS) is used to monitor and control the action of industry equipments and their associated devices. ICS is hardware and software integrated system and it uses three technologies called Distributed Control System, Programmable Logic Controllers, and Supervisory Control and Data Acquisition (SCADA). Using information technologies, the capability of control on input change by automating the monitoring and managing industrial processes like product circulation, handling, and production, etc (Rouse, industrial control system (ICS), 2016). In general, ICS systems are also called SCADA since both have the same usability in various industry resources. SCADA is one type of software application program for controlling instruments and conditions by collecting real-time data from remote locations and for controlling processes. Typically, SCADA or ICS is used in various divisions of national infrastructure, namely: mining, oil, gas refining and coal, power plants, transport and communication industries, water/waste water treatment, etc. (Rouse, SCADA (supervisory control and data acquisition), 2005). Although these systems suffer from IT security breaches and such breaches directly threaten the critical national infrastructure which means loss or compromise of availability, the integrity of dispatching nation's services and social and economic loss. For example, in December 2015, an intruder attacked the power stations of Western Ukraine and resulted in brownouts of 57 power substations of different parts of Western Ukraine. The attack is done by injecting malicious macro documents into the system using spear phishing emails and then performing distributed denial of service (DDoS) attacks with blank calls on customer service centers of various power stations. In the same year and same month, another reconnaissance attack happened on Calpine, America’s electricity generator company, and robbed remotely connected credentials of Calpine network, comprehensive engineering network diagrams, and information regarding 71 power stations nationally (Nigam, 2016). From this, we can realize that monitoring and controlling ICS system is very important. So, to prevent and mitigate such incidents and to ensure the security posture of ICS we proposed “Threat Intelligence and Machine Learning Approach to Industrial Control System Security”. The objectives of this approach include:

* To develop a strategy to collect and aggregate Threat Intelligence, Systems Operations, Network Behavior Data, and Human Behavior Data.
* To generate predictive model and detecting system anomalies by converting collected raw data into useful data as an input to machine intelligence system.
* To provide security analyst with assistance in recognizing or predicting systems anomalous behavior by designing and implementing machine intelligence visual system effectively and efficiently.

# 2. Literature review:

## 2.1 Machine Learning:

The MathWorks Team (MathWorksTeam, 2016) described two techniques of machine learning: supervised and unsupervised. While supervised learning uses known input and output data to generate a predictive model by using classification and regression techniques, the unsupervised learning type draws inferences by discovering underlying patterns in the input data by using clustering technique.

2.2 Machine Learning model notation: (Andrew)

= the column vector of all the feature inputs of the ith training example

= value of feature j in the ith training example

= target output that we are trying to predict

= training example

m = number of training examples

n = number of features

X = space of input features

Y = space of output values

h (hypothesis) = for given training set, to learn a function h: X->Y so that h(X) is a good predictor for corresponding value Y

J (cost function) = measure of the accuracy of proposed hypothesis (the difference between predicted and actual value)

= model parameter to define h

α = learning rate and constant lies between 0 and 1

= regularization parameter, controls the cost function from overfitting () or underfitting ()

***Regression*:** When the target variable that we are trying to predict is continuous, the learning problem known as a regression problem.

***Classification*:** When the target variable can take only a few number of discrete values, called it as a classification problem.

2.3 Gradient descent algorithm:(Andrew) The cost function indicates how well the data fits our defined hypothesis. To minimize the cost function error, estimating the parameters necessary. Gradient descent algorithms or batch gradient descent algorithms help to estimate and minimize cost function error.

***Algorithm:***

repeat until converges:

Where j = represents the feature index number, 0 to n

= learning rate. If too small, convergence is slow; if too large, cost function does not decrease or may not converge.

By setting our input values roughly in the same range, we can run the gradient descent algorithm efficiently. The reason is will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to optimum when the feature variable values are uneven. Feature scaling and mean normalization techniques might helpful.

***Feature scaling:*** It involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. The ideal ranges are -1 ≤ ≤ 1 or -0.5 ≤ ≤ 0.5

***Mean normalization:*** It involves subtracting the average value for an input variable from the values of that input variable resulting in a new average value for the input variable of just zero.

By implementing these two techniques, we can adjust the input variable with the following formula:

= average of all values of feature i

= range of values (max-min) or standard deviation

2.4 Linear regression algorithm**:** (Andrew) It tries to fit the data in the region of a straight line. It is effective if the training data has non-polynomial features. The disadvantage is computationally expensive for polynomial features.

2.4.1 For one feature variable***:***

Hypothesis:

Cost function:

Gradient descent:

Repeat until convergence: {

}

2.4.2 For multiple features***:***

=

Repeat until convergence: {

}

2.5 Logistic regression algorithm**:** (Andrew) It is used to classify examples into negative and positive classes based on decision boundary. For example, if ≥ 0.5, then example places into positive class otherwise, example places into negative class. It can be applicable to single classes or multiple classes classifier. The function of this algorithm is nonlinear.

Hypothesis or logistic function or sigmoid function:

***Cost function:***

***Gradient descent:*** repeat: {

}

2.6 Neural networks**:** (Andrew) It is another model to train data in a nonlinear fashion. Also, it is an effective algorithm for classifying data into multiple classes. The feature sometimes called as bias unit and it uses the same logistic function which is used in logistic regression, sometimes called it as the activation function. The parameters also called as weights. The following is the simple representation of this model.

Usually, the function model is represented in the form node layers. The input feature nodes are treated as input layer and the layer at which hypothesis function outputs called it as output layers. The intermediate layers between the input layer and output layer are called hidden layers and its nodes are called activation units.

***Notation:***

= “activation” of unit i in layer j

= matrix of weights controlling function mapping from layer j to layer j+1

L = total number of layers in the network

Sl = number of units (excluding bias unit) in layer l

K = number of output units/classes

Note: if network has units in layer j and units in j+1 layer, then will be of dimension

Activation function:

2.6.1 Forward propagation**:**

where temporary matrix and = x (input feature matrix)

**Cost function:**

(the i in triple sum does not refer to training example i)

2.6.2 Backpropagation**:** It is a neural network terminology and the functionality is similar to gradient checking, i.e. to minimize the cost function.

***Algorithm:***

Given training set { , , ……… }

* Set for all (l,i,j)
* for (training example) i = 1 to m {
  + set
  + perform forward propagation to compute
  + using , compute
  + compute using

}

* if j=0

2.6.3 The best approach to training neural network**:**

1. First, pick a network architecture; choose the layout of the neural network by determining number of input and output units, number of hidden units per layer, and total number of layers to include in the network.
2. After picking a neural network, randomly initialize the weights.
3. Implement forward propagation to get hypothesis for each feature.
4. Implement the cost function.
5. Implement back propagation to compute partial derivatives or minimize the cost function.
6. Use gradient checking to confirm that back propagation is working correctly, then disable gradient checking.
7. Use gradient descent or built-in optimization function to minimize the cost function with the weights in theta.

2.7 Support Vector Machine (SVM)**:** (Andrew) A robust algorithm tries to separate data or classifies data with a large margin. It mainly uses with various kernels. Kernels draw new features from x based on landmarks (l(i)) with similarity function (similarity (x, ) = where ).

***Hypothesis***:

= 1 if

= 0 otherwise

***Cost function***:

If y=1 (wants )

If y=0 (wants )

where C is regularized parameter (reciprocal of and act as decision boundary.

2.7.1 SVM with kernels (Gaussian Kernel)**:**

Choose landmarks, , ,……., from given training examples, , , ………, then compute new features(..,) using similarity function. With these new features, predict y=1 if and compute cost function

2.8 K-means clustering algorithm**:** (Andrew) This algorithm falls under the unsupervised learning algorithm where output labels or variables are unknown. It groups the given data in the form of clusters. Market segmentation, social network analysis, organize computing clusters, and astronomical data analysis are some applications of this algorithm.

***Notation:***

= index of cluster (1,2,3, …, K) to which example is currently assigned.

= cluster centroid k

= cluster centroid of cluster to which example has been assigned.

***Algorithm:***

Input: K (no. of clusters), Training set {}

Randomly initialize K cluster centroids .

Repeat {

for i =1 to m

= index (from 1 to K) of cluster centroid closest to

for k = 1 to K

= average (mean) of points assigned to cluster k

}

***Optimization objective:***

2.9 Anomaly detection algorithm**:** (Andrew) It is used for density estimation or finding an anomalous sample from given data set by modeling P(x) from data. It helps to raise a flag when abnormal behavior occurs. Fraud detection, testing manufacturing equipments, and monitoring computers in a data center are the major applications of this algorithm. It uses the concept of Gaussian distribution.

***Algorithm:***

* Choose features that might be indicative of anomalous examples.
* Fit parameters
* Given new example, compute p(x):
  + =
* Raise anomaly if p(x) <

Using F1 score metric can evaluate the algorithm accuracy on cross-validation and test set data.

2.9.1 Anomaly detection vs. Supervised learning:

|  |  |
| --- | --- |
| **Anomaly detection** | **Supervised learning** |
| Appropriate if few number of anomalous examples available or large number of non-anomalous examples available.  Effective if a high number of anomaly types present; if future anomalies look like nothing and have a small number of anomalous examples, modeling with non-anomalous examples is appropriate.  Ex: Fraud detection, manufacturing, monitoring machines in a data center, etc. | Appropriate if a large number of anomalous and non-anomalous examples exist.  Requires enough positive examples to get a sense of what positive examples are like, future examples likely to be similar to the training set.  Ex: Email spam classification, weather prediction, cancer classification, etc. |

3. Threat Intelligence (Chismon & Ruks, 2015):

The Intelligence Community (IC) conveys Intelligence as a tradecraft, relates to techniques, methods, and technologies that can be connected in all industry vertical’s and used in spying. According to IC, intelligence accumulation comprises five core disciplines:

**i.** **Human Intelligence** collects data from human sources by citing, listening, or through an action.

**ii.** **Open Source Intelligence** explores, and improves public data from sources such as TV and radio broadcasting, daily newspapers, books and internet, etc. using data mining and advanced search strategies.

**iii.** **Signals Intelligence** is a merge of electronic intelligence and communication intelligence and it accumulates and exploits signals transmitted from communication systems, radar and weapon systems.

**iv.** **Imagery Intelligence** acquires geospatial data and is undertaken by a range of terrestrial, airborne or satellite based collectors.

**v.** **Measurement & Signature Intelligence** determines objects or events by accumulating data from radars, lasers or sensor devices and it can distinguish a person, place, or object. The acoustic, nuclear, chemical and biological intelligence are also part of this intelligence.

Threat Intelligence is a complete knowledge of threat capabilities, infrastructure, motives, goals, and resources of an organization and that contains mechanism, indicators, implications, and suggestions to provide decisions to nullify threats. The threat data can be obtained from internal and external sources. Internal sources mean leveraging threat intelligence from an enterprise network like log files, alerts, and incident response reports; they can detect and prevent hazards. External sources are open source intelligence namely, security researcher or vendor blogs, publicly available reputation and block lists provide indicators for context and identification of a threat.

Such commercial sources of threat intelligence contain threat intelligence feeds, structured data reports (STIX), unstructured reports (PDF and Word documents), emails from sharing groups, etc. Threat intelligence tolerates an organization to generate the most relevant and accurate threat profile and to rate and rank the value of sources of threat intelligence. This threat data gathered by using either holistic (considers all aspects of organization threat management) or focused (concentrated on assets of organization network) approach.

3.1 Intelligence lifecycle (Chismon & Ruks, 2015):

The development of threat intelligence involves a process of converting raw data into intelligence data known as intelligence life cycle and it can be done in five phases.

**Phase1:** Planning, Requirements, and Direction as per consumer needs. Identifying requirements refer to what consumers want to know and what the Threat Intelligence outcome, etc. For collecting intelligence, Planning and Direction provide management from identifying and prioritizing requirements to the final product of Threat Intelligence.

**Phase2:** Collecting the expected data from a variety of sources such as internal and external sources, core intelligence disciplines. It should fulfill the requirements.

**Phase3:** Processing the gathered data by synthesizing data into standard/structure format suitable to do a detailed analysis. For that, the analysts analyze the gathered data to identify threats and look for countermeasures to improve threat intelligence.

**Phase4:** After analysis, intelligence product is created and disseminated to stakeholders to take appropriate measures.

**Phase5:** Evaluating outcome product to ensure it meets all requirements. If requirements met, add additional requirements to develop new product or else the cycle process starts again from failure occurred place.

3.2 Types of Threat Intelligence (Chismon & Ruks, 2015):

Based on the consumers, Threat Intelligence is categorized into four subtypes.

3.2.1 Strategic Threat Intelligence**:**

This type of intelligence is used by top level strategists, the board of trustees, or persons of an organization to operate high-level risks and their probabilities instead of technical aspects. This intelligence is used to aid strategist’s ability to grasp current risks and recognize forthcoming unknown risks. Additionally, it advises the board when making strategic business decisions and provides understanding of consequences of those decisions. A collection of strategic data is a big challenge to strategists and it requires socio-political mindset. Usually, the organizations buy it from strategic intelligence providers such as Open Source Intelligence, Security Industry White Papers, Human Communications (Chismon & Ruks, 2015).

3.2.2 Operational Threat Intelligence**:**

This sort of Intelligence indicates type of the attack, the identity, and ability of the attacker and provides an early sign of happening attack and decreases by cutting attack paths or hardening services. This intelligence is used by higher-level security staff, such as security managers or heads of incident response. The requirements to produce this intelligence are related to a specific group and ensure whether these requirements appropriate or not. This data is gathered by recruiting human sources into a group. For that these group monitors on recurring activity associated with attacks, observing chat room discussions, and examining social media networks using scripts or code to identify interesting messages helpful to find indications of an attack (Chismon & Ruks, 2015).

3.2.3 Tactical Threat Intelligence**:**

Tactical Threat Intelligence is a standout and most useful intelligence to understand how the attackers try to attack and then applying that concept to mitigation or recognition procedure. It deals with data associated with tactics and it is done by tools and methodologies of threat group. It is mostly used by defenders such as architects, administrators, and security staff. The requirements to this intelligence involve examining and investigating the tactics used by the threat groups and setting requirements based on consumers. Usually, the data is gathered from attack campaign reports, malware samples, incident response reports, etc (Chismon & Ruks, 2015).

3.2.4 Technical Threat Intelligence**:**

Technical Threat Intelligence mainly focuses on technical tools, command and control (C2) channels, and infrastructure which are used by an intruder as well as specific indicators. For example, use of a malware piece can be a tactical intelligence whereas indicator against a particular malware would be technical intelligence. Familiar indicators for this intelligence consist of MD5 malware or report draws, phishing electronic mails, IP addresses for C2 endpoints or domain names of C2. By incorporating these indicators in defensive infrastructures (like firewalls, mail filtering devices, endpoint security solutions and so on.) organizations look to distinguish attackers. Moreover, historical attacks find outs by observing earlier connections or binaries (Chismon & Ruks, 2015).

# 3.3 Threat Intelligence Maturity Model:

Maturity model helps to determine the organization security posture and guides to enhance security posture by reviewing each stage and learning about resources, organization structure, and technologies needed to attain strategic processes and operationalized organization threat intelligence. The model provides direction to know organization capabilities, risks, and exposures at each level. Maturity model consist of five levels (ThreatConnectTeam, Maturing A Threat Intelligence Program, 2015).

1. **Level 0:** In an organization, while gathering the data the team may make mistakes which hold Security Information and Event Management (SIEM) observing external problems due to lack of awareness of unstructured data, invalid data. At this level, the risks are uncountable due to manual work process which focuses on historical threat without considering adversaries which vary with time.
2. **Level 1:** Correlating internal threat data with external feeds adds a part of intelligence to an organization. The combination of level 0 and level 1 gives better threat intelligence. Although at this level, organizations have several difficulties like lack of finding the threat actors, incident place, and pattern behavior because of SIEM approach.
3. **Level 2:** At this level, the limitations of level 1 can be covered in the means of interacting with external sources by building and defining processes as well as adding Indicators of Compromise (IOC). Intelligence at this level fails to address if threat intelligence requirements exceed its capacity due to rapid change of threat sources. By deploying analytical threat intelligence program, resources can mitigate that impact.
4. **Level 3:** Organizations possess better threat intelligence processes and drive connection with outside partners to protect adjacent organizations networks. Organizations begin to equip the threat intelligence platform, which automates the workflow of multiple data and reduces the labor-intensive process but it is not completely automated to read all formats of data.
5. **Level 4:** At this level, organizations are highly secured and have greatly defined and established threat intelligence platforms and threat intelligence program that consists of processes, workflows to actionable intelligence, and quick response.

# 

# 3.4 Threat Intelligence Platform (TIP):

A platform which supports all types of security teams from top level, C-series (CEO, CSO or CISO) to bottom level security team is termed as threat intelligence platform and used to do threat analysis, network defense by blocking and tackling, also strategic decision-making and development of processes to handle outside source’s feeds and intelligence. It should provide the intelligence lifecycle management for threat intelligence program. In general, SIEM used to aggregate the data of events that are identifying incidents, threat relevance, etc. Integrating SIEM and all other internal application sources like correlation engine with TIP can be a most powerful strategy to enrich threat intelligence and to manage knowledge vastly (ThreatConnectTeam, Threat Intelligence Platforms, 2015).

TIP is more appropriate for any organization irrespective of the level of threat intelligence capability. This intelligence can be achieved by three functionalities of TIP: ‘Aggregate’, which deals with collection, processing, exploitation phases of intelligence cycle; ‘Analyze’ function, which handles analysis and also production, planning and direction phases of intelligence cycle; ‘Act’ function, which manages dissemination, feedback, and requirement phases of intelligence cycle. The process automation of TIP achieved by integrating TIP with an organization network using strong Application Programming Interface (API). One case study by Phenomenon Institute unveils sharing TI knowledge, indicators, and historical data through open sources (blogs and reports), spear phish email, community threat groups tremendously increase the security posture of the organization (ThreatConnectTeam, Threat Intelligence Platforms, 2015).

## 3.5 Cyber Kill Chain in ICS:

The two stages of cyber kill chain process help defenders to identify and distort attackers and maximize the cost of attack before they successfully attack (Assante & Lee, 2015).

**Stage-I of cyber kill chain:** (Assante & Lee, 2015)

At this stage, the intruder tries to acquire ICS system information, learn that system, and introduce mechanisms against internal system safeguard parameters or entry into a production environment. It consists of several phases showed in figure 1.

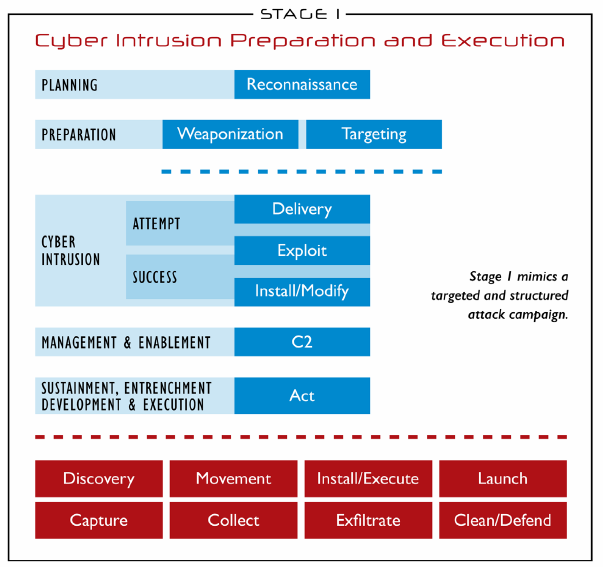


Figure : ICS Cyber Kill Chain Stage 1

Source: retrieved from the article: the industrial control system cyber kill chain (Assante & Lee, 2015)

**Planning phase:** At this phase, attacker performs reconnaissance. Reconnaissance is a process of retrieving information by means of any detection approach. Also, it can be research on target either through online like Google data search engine or offline like data of public announcements and social media profiles. The reconnaissance mainly targets ICS vulnerabilities and it provides a way to understand process and operating model of the system to exploit it. Passive reconnaissance methods (footprinting) are often used by attackers to produce data related to the target (Assante & Lee, 2015).

**Preparation phase:** Weaponization or targeting can take place in this phase. Weaponization is nothing but adding exploit content in non-harmful documents such as PDFs and word documents, for performing next attacking step. If an intruder identifies through script or tool any potential victims for exploitation, targeting action takes place, which is a process of analyzing, prioritizing targets, and preparing corresponding actions to target (Assante & Lee, 2015).

**Cyber Intrusion Phase:** This phase is crucial to obtain initial access. Intrusion is the probability of success or fails to obtain access to the defender’s network or system. Delivery and exploit are the key parts in the intrusion. Delivery step tells what type of method can be used to get into defender’s network, like sending weaponized PDF through emails. The exploit step indicates that the intruder performs malicious actions, like to exploit a vulnerability when a weaponized document opens or exploitation of existing procedure to get into the network (Assante & Lee, 2015).

**Management and Enablement Phase:** Once cyber intrusion went to success, using methods like a connection to the formerly situated capability or hijacking trusted communication, intruder establishes single or multiple command and control (C2) paths by concealing outbound and inbound traffic. These multiple C2 paths assure an uninterrupted connection even if one of the paths is detected or removed. After that, the attacker is capable of achieving their desired goals (Assante & Lee, 2015).

**Sustainment, Enforcement, Development, and Execution phase:** Here, adversary creates a list of end goals and acts on it. Some of the common adversary end goals are “finding new systems or data, lateral movements around the network, adding and executing capabilities, launching of those capabilities, apprehending transmitted communication. This is a crucial and critical phase for handling actions at stage-2 of cyber kill chain where all “ICS data and industrial process, engineering, and operations presented on the internet oriented corporate or enterprise network” (Assante & Lee, 2015).

**Stage-2 of cyber kill chain:** (Assante & Lee, 2015)

At stage-2, to perform an attack against ICS, adversary needs knowledge acquired from stage-1 to generate and test capability. However, most of the stage-1 adversary operations often lead to unintended attack due to the existence of sensitive equipment. Also, it poses unforeseen consequences, which is a severe risk a nation-state cyber operation. In contrast, the attacks in stage-2 are intentional and are showed in figure 2.

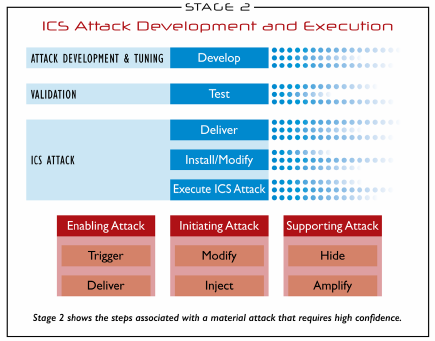


Figure : ICS Cyber Kill Chain Stage 2

Source: retrieved from the article: the industrial control system cyber kill chain (Assante & Lee, 2015)

**Attack Development and Tuning:** In this phase, with the help of data exfiltration, the attacker generates new capability with respect to the desired goal and considering particular ICS implementation. The detection of adversary development and tuning is also difficult because of extended development and testing time to lessen the considerable gap between stage-1 and stage-2 operations. It could also be difficult because either these system owner or operator are less capable of dealing with adversary development (Assante & Lee, 2015).

**Validation:** Here, the intruder tests the developed capability on similarly configured systems to find out any impact. Increased network scanning for the denial of service to systems and the amount of testing required to finalize that the scanning can deny service to the systems are some simple impacts. The impact will be more when the adversary performs more testing to gain physical ICS equipment and software components (Assante & Lee, 2015).

**ICS Attack:** This is the last phase in the ICS cyber kill chain where the attacker will send validated capability, install it, or alter previously defined system functionality, and then carry out the attack. “Preparatory or concurrency” are the main facets of attack, which belong to attack categories of “enabling, initiating, or supporting” to accomplish desired effect. These are triggering conditions to take control over process element, initiating process set points and variables alteration or supporting the attack. The total complexity of attack completely depends on various factors such as the level of system security, the level of monitoring and controlling the process, the safety design and control, and impact level (Assante & Lee, 2015). There are several methods to attack ICS environment and these are shown in figure 3.

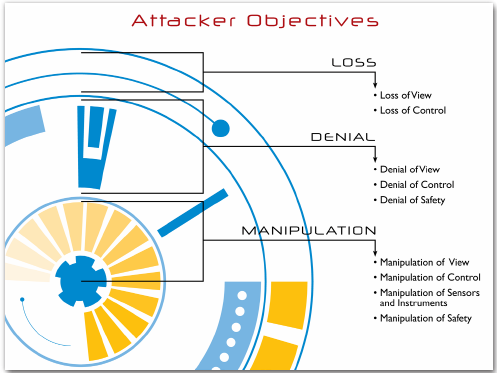


Figure : Attacker Objectives

Source: retrieved from the article: the industrial control system cyber kill chain (Assante & Lee, 2015)

3.6 Case study – Cyber kill chain process in the Ukraine Power Grid attack**:**

**ICS Cyber Kill Chain Mapping-Stage 1:** (Case, 2016)

The first step in the attack process is reconnaissance. The adversary performed reconnaissance before targeting energy companies and picked three organizations as targets after analyzing levels of automation in their distribution and capacitating the remote opening of breakers in more than one substation. The strong evidence to occur reconnaissance is highly coordinated between the targeting and final attack plan. Secondly, they weaponized Microsoft Office documents by inserting BlackEnergy 3 malicious codes within the documents. Next, throughout the cyber intrusion phase, they transferred a malicious Office file by means of emails, i.e. spear phishing to everyone in the administrative or IT network of that company. When these were opened by the user, it showed a macro enabling pop-up window. Once accepted, malware of that macro exploits macro functionality in Office documents and then installs BlackEnergy 3 on attacking the system but not catch any vulnerability exploitation codes. Essentially, they placed macro functionality to use throughout the system(Case, 2016).

Upon installation, the adversary enabled communication to the infected system by establishing a connection with C2 IP addresses using BlackEnergy 3 malware, which allows acquiring system information and access to the environment. In this case, the intruder accessed the environment six months before the actual attack happened and performed required actions to access the target uninterruptedly. In these actions, the adversary is gathered credentials at initial foothold for the systematic takeover of IT systems and remote connections and left immediately away from initial footholds and vulnerable C2s to ensure to enter target system as authorized users. With this information, the intruder easily recognized virtual private network (VPN) connections and links from the business network to ICS network. Once entered the network where SCADA dispatch workstations and servers existed, the intruder performed desired activities but technically different minutia between three affected oblenergos. Then they discovered a network connected to an Uninterrupted Power Supply (UPS) and made power outage by reconfiguring that network(Case, 2016).

**ICS Cyber Kill Chain Mapping-Stage 2:**

In the attack development and tuning phase of stage 2, the intruder performed the develop step minimum in two ways. First, they studied the way to interact with three different DMS environments with the help of native control available in the system and operator screens. Secondly, the adversaries developed malicious firmware for the Serial-to-Ethernet devices and then uploaded it in multiple sites. This malicious firmware ensures quick and predictable execution. At validation phase before deployment, attackers performed a test on developed capabilities professionally rather than just relying on luck throughout the rest of the attack. In the course of ICS attack, attackers used remote administration tools on operator workstations to “deliver” themselves into the environment for direct communication with the ICS components. Then attackers finished “install/modify” step by installing tailored software, called “KilDisk” over the environment. With that, it ensures modification to UPS would be possible and it is ready for the attack. With same modification act attackers also took control over operator workstations and block the operators completely out of their systems(Case, 2016).

Finally, the intruders executed the ICS attack using HMIs in the SCADA environment to open breakers. Consequently, minimum 27 workstations of three energy companies went to offline. Meanwhile, the intruders also uploaded malicious firmware into serial-to-ethernet gateway devices, which ensures blocking remote commands to get back to an original online status even if operator workstations recovered. At the same time, adversaries performed remote telephonic denial of service on company’s call center with thousands of calls to prevent customers to report service outages. The main purpose of denial of service attacks is to frustrate customers by not knowing the reason behind the outage. However, in the entire attack process, BlackEnergy 3, unreported backdoors, KillDisk, and malicious firmware are part of the attack and each one is not responsible for the outage. The outage happened due to manipulation of the ICS itself and complete loss of control because of direct intuitive operations by the intruder(Case, 2016).

# 3.7 Cyber Kill Chain with Threat Intelligence Platform:

To improve the computer network defense and mitigate the risks, the organization should always try to place cyber kill chain framework process into their network. The benefits of this framework are organization which can reduce risks, erect true resilience, improve the accuracy of measure results, and develop greater communication. In addition, TIP can help to augment these benefits and achieve those benefits. TIP could apply the cyber kill chain process in seven ways. They are prioritizing the incoming sensor alerts from various sources; prioritizing escalation for providing better interaction for analyst with intrusion report of alerts to know the level of impact; prioritizing investments for building future investments by representing attributes (detect, deny, deceive, degrade, etc.) of protection at each stage of kill chain process in visual datagram; improving effectiveness through early blocking of attacks or intrusions; measuring resilience with complete understanding of located defenses at every stage of cyber kill chain; measuring analytic completeness by focusing on occurrence of attack, seeing possible outcomes if fail of synthesization beside to recognize and stop the attack; arranging track campaigns by identifying and grouping same type of intrusions into campaigns such that organizations can undoubtedly prioritize and measure the adversaries (Martin, 2015).

# 3.8 Threat Indicators:

Threat indicators are information that illustrates or recognizes the malicious reconnaissance, the procedure to exploit the vulnerability or defend security control, harms caused by threats, activities of the legitimate user when accessing the system or information (Team, 2015). The common threat indicators: indicators for cyber threats namely, phishing and spear phishing, malicious code, weak and default passwords, unpatched or outdated software vulnerabilities, removable media; network indicators like C2, Havex, etc.; malware indicators; email indicators (Team, 2015).

# 3.9 Indicators of Compromise (IOC):

IOC is often referred to as scientifically investigated data for an attack. It helps to notate whether or not malware items exist in an environment with the certain degree of confidence and work as backend answer for data-stealing attacks. In general, IOC’s are described with some standard formats like Open IOC, Structured Threat Information eXpression (STIX), Cyber Observable eXpression (CybOX), Trusted Automatic eXchange of Indicator Information (TAXII), Incident Object Description Exchange Format- RFC 5070, YARA, etc. (Andress, 2015).

3.9.1 IOC types: The following are the typical indicator types of IOCs (Josh, 2015):

3.9.1.1 Email indicators**:** An email associated with suspicious activity. Email IOC can be identified by some of the following listed ways (Josh, 2015):

* ***Senders email address and email subject:*** The aim of threat actors is forcing the public to open socially engineered emails by creating emails from addresses which are publicly recognizable individuals, inserting popular pictures, emphasizing current ongoing events, and so on.
* ***Attachments and links:*** These are harmful and use in spear phishing emails and campaigns. For monitoring and tracing purposes these attachments are continuously reused.
* ***X-forwarding IP address:*** It is a header field and it is not directly determined client original IP address rather recognizes the address when client connecting web server through a HTTP proxy or load balancer so that it allows new insights regarding attack process against an industry. To perform an attack or to send social engineering emails, compromised servers and attack infrastructure are reused.
* ***X-originating-IP address:*** It is a header field and determines client IP address when connecting to the mail server. The appearance of this field depends on mail server being used. Closer observation in this field provides additional insights against threat actor’s infrastructure.

3.9.1.2 Network Indicators**:** These IOCs unveiled in following ways (Josh, 2015).

* ***URL:*** Uniform Resource Location (URL) summarize the online location of a file or resource. The attackers used on C2 (Command and Control) server and for delivering malwares through links.
* **Domain names:** Domain indicates a domain name for a website or server. It encompasses a series of hostnames. It is used for C2, placing malwares in data exfiltration sites and sending malicious links by means of socially engineered email attacks.
* ***Hostname and URI:*** The hostname reveals hostname for a server located within a domain whereas Uniform Resource Indicator (URI) describes the explicit path to a file hosted online.
* ***IP addresses:*** The IPV4/IPv6 address indicates the online location of a server or another computer. These IP addresses help to stop distributed denial service attacks from compromised servers, botnets, and systems. It will not work when actor moved from compromised server to another other and due to the growth of cloud-based hosting services.
* ***User-agent strings:*** Helps to determine the operating system of the computer, type of browser and some particular instructions which could be allowed on web pages and providing correct data to the client.
* ***CIDR indicator:*** Classless Inter-Domain Routing (CIDR) address, which describes both a server's IP address and the network architecture (routing path) surrounding that server.

3.9.1.3 Host-based indicators**:** These are obtained by analyzing computers which are prone to damage and this analysis done within in an organization. The following are some ways to identify host-based iocs (Josh, 2015):

* ***Filenames and File hashes:*** The names consist of malicious executable files and decoy documents. Using hashes, these files can be investigated. The file hashes include MD5, SHA1, SHA256, PEHASH, IMPHASH, and so on. All these are their respective formats of hash that summarize the architecture and content of a file. The file path indicator gives a unique location in a file system.
* ***Registry keys:*** These are generated by malicious code and to preserve these, some of the keys get altered in computer’s registry. It is commonly used for creating Trojans.
* ***Dynamic link libraries (DLL):*** These are files intruder trying to replace windows system files at the time of windows startup procedure. Once successfully loaded during startup procedure the windows start executing its payloads.
* ***Mutual Exclusion (mutex):*** The name of a mutex resource describing the execution architecture of a file and limits access to the resource. The intruders use this indicator to confirm whether the host is successfully infected by one of the malware instances.

3.9.1.4 CVE indicator**:** Common Vulnerability and Exposure (CVE) entry describing a software vulnerability that can be exploited to engage in malicious activity.

3.9.2 IOCs in SCADA with CIMPLICITY HMI case study**:**

Some of the typical IOCs relevant to the GE intelligent platforms CIMPLICITY HMI solution suite are listed in below. The attackers are trying to place “.cim and .bcl” files into CIMPLICITY installation directory using a %CIMPATH% environmental variable on the machine (Kyle & Jim, 2014). Sandworm security team focused on one of the C2, 94[.]185[.]85[.]122 and identified a file related to GE CIMPLICITY software object oriented file called CimEdit/CimView, which is used to regulate the SCADA components. That recognized file is named “config.bak file (SHA1 hash: c931be9cd2c0bd896ebe98c9304fea9e)”. The config.bak consists of two events: “OnOpenExecCommand”, which is used to drop “default.txt” and; “ScreenOpenDispatch”, which is a subroutine that initiates the downloading of the “newsfeed.xml” file. However, this mechanism does not have a capability to exploit the vulnerability. Further, additional IOC is identified “devlist.cim (MD5: 59e41a4cdf2a7d37ac343d0293c616b7)” and treated as “Cimpack Design Drawing File”. The deault.txt file allows the execution of the “Flashplayerapp.exe” file, which in turn executes the commands such as “exec, lexec, die, getup, turnoff, chprt”. Additional script file is called “shell.bcl”. In SCADA, .bcl files are used to automate some functions. Based on the text in shell.bcl, the xv.exe is much more susceptive to exploiting system vulnerabilities. Another IOC in C2 is the Spiskideputatovdone.ppsx (MD5: 330e8d23ab82e8a0ca6d166755408eb1). This indicates Russian deputy list and embedded to an email address oleh.tiahnybok@vosvoboda.info as reported by on a VirusTotal submissions. This file is a PPSX file that downloads/loads “slide1.gif and slides.inf(MD5:8313034e9ab391df83f6a4f242ec5f8d)”. The “slide.inf” changes the name of “slide1.gif” to “slide1.gif.exe” and adds a registry entry:

“HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce Install=”{dir}\slide1.gif.exe”.

The Slide1.gif.exe (MD5: 8a7c30a7a105bd62ee71214d268865e3) drops FONTCACHE.DAT (MD5: 2f6582797bbc34e4df47ac25e363571d), which is black energy bot and executes commands such as “delete, ldplg, unlplg, update, dexec, exec, updcfg” (Kyle & Jim, 2014).

3.10 MineMeld Threat Sharing Tool**:**

MineMeld, developed by Palo Alto Networks, is an open source tool that works on aggregation, administration, and distributing threat intelligence. The tool can easily integrate all intelligence feeds which includes outputs from other threat platforms from various sectors, namely public, private, and commercial. Then unified feeds fully capable security prevention controlled framework and supplies it to Palo Alto network or any security application. This tool can be deployed either using API, Web UI or pre-built virtual machines so that as per environment organizations can easily customize input processing and output of information (Scott, 2016).

The MineMeld consists of three blocks and it is shown in figure 4. The first block selects miners, which are threat intelligence sources like indicator feeds or service oriented threat intelligence but may require a subscription. For example, AutoFocus threat intelligence service, which is offered by Palo Alto networks; the Second block selects processors, which absorb indicators from miners and conducts the action on that indicator. This action varies with selected processor. For instance, The MineMeld processors perform wrangling on data to absorb unique and desired indicators from the bulk. It is also controlled by the end user ensuring which miner can do filtering and aggregation; Third block selects the desired output. This tool, after processing the indicators, automatically forwards to selected output like “Palo Alto Networks dynamic address group, external dynamic list, or a TAXII feed“. By configuring MineMeld, one can also disseminate indicators from multiple processors to multiple selected interested outputs (Patricia, 2017).

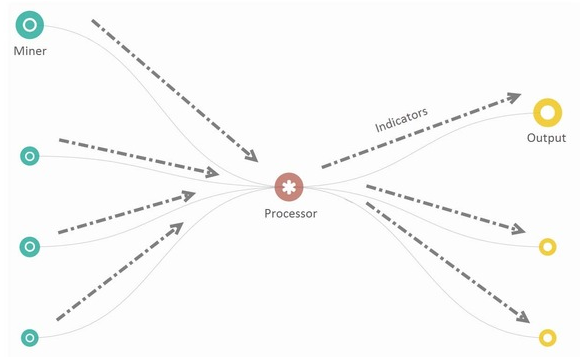


Figure : MineMeld working functionality

*source: Retrieved from paloalto networks website* (networks, 2017)

4. ICS System Analysis & Design and Implementation**:**

The threat data can be generated by three behaviors: user behavior, network behavior, and system behavior. The operational threat intelligence is obtained by analyzing these three behaviors. The following table 1 describes the analysis on three behaviors and categorizing the activities based on data behavior.

|  |  |  |
| --- | --- | --- |
| **User Behavior Analytics** | **Network Behavior Analytics** | **System Behavior Analytics (SCADA)** |
| * Logon/Logoff activities * Software installation * Modification of system file/registry * Browsing activities * File transfer activities/Data exfiltration * Threat Intelligence * Email activities * Network connection activity (wireless/mobile) * Privilege escalation * Account takeover | * Number of packets received/sent * Geolocation of packet source * Types of packets (protocols) * Size of packets * Time of network activities * Threat intelligence * Status of routers/switches * Malware detected * Firewall status | * Water pump statistics * Water tank levels * Alarm statuses * Equipment statuses * Motor speed * Data transfer rate * RTU statuses * Communication statuses * System program modification * HMI access/modification * Historian access/modification |

Table 1:Categories of Behavior Analytics

4.1 Data Collection and Monitoring**:**

4.1.1 Data Collection Elements:

All data was collected through Splunk from local Splunk Forwarder services, except for the Palo Alto Firewall's sys logs. During the collection period, no other systems were used to avoid altering the firewall logs and to create a more accurate baseline collection. All events are time-stamped.

**i. WinHostMon (windows host monitoring):** A Splunk collector for Windows hosts that monitors several different data sources. Activated by editing the Splunk Forwarders' “inputs.conf “ file. The sources are included in below table, but not limited:

|  |  |  |
| --- | --- | --- |
| **Windows Services** | **Windows Processes** | **Windows Disk** |
| * Names and descriptions of services * State (started or stopped) * Path of service * Start mode (auto, manual, disabled) | * Names and descriptions of processes * Process command-line * Path of process * Process ID | * Name (drive letter) of disk * Filesystem * Free and total storage space * Drive type (CD-ROM, fixed, removable) |

**ii. EventMon (Event Monitoring):** A Splunk collector for Windows hosts that monitors specified logs from the event viewer. It collects the following:

* Severity of log
* Host's computer name, domain, and network address
* User logons, logoffs, logon types
* Caller process IDs
* Event descriptions and event codes
* Logon sources

**iii. RegMon (Registry Monitoring):** A Splunk collector that monitors specified Windows registry hives. This collection was set up to only forward registry changes associated with the HMI, as this collection can generate a lot of data. The Splunk collects below listed data:

* Registry key deletion, creation, and modification
* Application associated with change
* location of the change in the registry

**iv. SEP (Symantec Endpoint Protection):** Monitored by exporting SEP's logs as text files and forwarding them through Splunk:

***Packet Logs:***

* Network traffic to and from hosts running SEP
* Protocol
* Local and remote ports for packet source and destination
* Rule (allowed/blocked)
* Direction
* Application sending/receiving

***Security (no logs generated during collection):*** Alerts triggered by potential attacks.

**v. PAN (Palo Alto Network):** Palo Alto firewall sys logs, parsed by the Palo Alto Splunk app. More specific data can be gathered, including URLs visited, with an active license for the firewall. Splunk collects following data:

* Applications passing data through firewall
* Categorized applications (Business, VOIP, Encrypted Tunnel, etc.)
* Application behavior (Evasive, can send files, used by malware, etc.)
* Known vulnerabilities of applications

**vi. Perfmon (Performance Monitoring):** Splunk collector that monitors logs created by Windows Performance Monitor. Counters for the desired data must be activated in Perfmon.exe first. For this collection, only the network adapter was monitored to bytes sent and received. It lists following details:

* Network adapters
* Count of bytes sent and received

### 4.1.2 Data Collection and Aggregation:

The baseline and test data collection duration are exactly 24 hours. For data collection purpose, the user actions like login/logout, file transfers, interactions with HMI, etc. are automated with an existed python script. Using batch script run the python script from 6 AM to 2 PM to simulating a workstation shift. With the help of Splunk server and local Splunk Universal Forwarder services monitored two specific hosts, namely HMI server and Historian server, and gathered data from different data source elements listed in data collection elements section. At the point of data collection, no systems were utilized to abstain from changing the firewall logs and to make a more precise baseline data collection. All events are time-stamped (Francia, 2017). The data collection, monitoring, and visualization system is shown in figure 5.

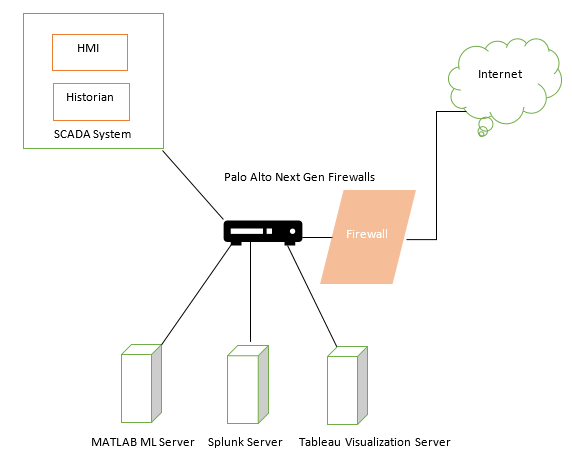


Figure 5: Data collection, monitoring, and visualization system

After that performed data aggregation by exporting collected data from Splunk server. During aggregation, data set is examined and explored and selected the data fields/ features/attributes, which are listed in a table (available in the appendix).

4.1.3 Threat Indicators collection from Alien Vault**:**

AlienVault developed Open Threat Exchange (OTX) platform to exchange open threat data in order to detect and respond to threats rapidly so that organizations can enhance their security posture. OTX allows us to create/edit/share pulse, which is the composition of IOCs. These IOCs guide users to safeguard against adversaries by illustrating an actor, threat campaign, or any activity. By simply subscribing desired pulses, users can export and integrate into their application. One serious feature of OTX is exploring and scrutinizing every indicator of a pulse, i.e. it provides all details of that indicator. For instance, if indicator type is IP or domain name, it will present details such as “URLs, passive DNS data, IP reputation data, and malware samples which are known to IP address” (Jaime, 2015). In this project, using OTX Python SDK collected ICS or SCADA related indicators of compromise by writing a Python script (available in the appendix). A few of the IOCs are listed in table 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Indicators of Compromise | Possibility of being Benign | Attack Category | Attack Category Name |
| A change has been made to the Windows Firewall port exception list | **YES** | 2 | System |
| Attempt to install service | **YES** | 2 | System |
| Audit policy change | **YES** | 2 | System |
| Data Exfiltration | **YES** | 2 | System |
| Event code indicates failed logon and account locked | **YES** | 2 | System |
| Excessive disk space used | **YES** | 2 | System |
| New removable drive types | **YES** | 2 | System |
| Register set value-reported as the winnt malware | **NO** | 2 | System |
| Special privileges assigned to new logon | **YES** | 2 | System |
| The audit log was cleared | **YES** | 2 | System |
| The event logging service has shut down | **YES** | 2 | System |
| Unable to log events to security log | **YES** | 2 | System |
| Windows firewall notification blocked | **YES** | 2 | System |
| Buffer Overflow Vulnerability | **NO** | 3 | CVE |
| DATAC Realwin SCADA Server Initialize buffer Overflow Vulnerability | **NO** | 3 | CVE |
| Unusual application: SecEdit | **NO** | 3 | CVE |
| Hash Signature | **NO** | 4 | Threat |
| Malicious IP | **NO** | 4 | Threat |
| Malicious Host/DNS | **NO** | 4 | Threat |
| SCADA System Format String Vulnerability | **NO** | 4 | Threat |
| Malicious application (malware) | **NO** | 4 | Threat |
| Improper Access Control CIP on Control Logix Change IP | **NO** | 4 | Threat |
| Improper Access Control CIP on Control Logix Reset | **NO** | 4 | Threat |
| Improper Access Control CIP on Control Logix Replay | **NO** | 4 | Threat |
| Improper Access Control CIP on Control Logix Firmware Upload | **NO** | 4 | Threat |
| Change password attempt | **YES** | 5 | User |
| Logon Failure - Unknown user name or bad password | **YES** | 5 | User |
| User account created | **YES** | 5 | User |
| User account enabled | **YES** | 5 | User |
| User rights assigned | **YES** | 5 | User |
| Web browsing on unusual port | **YES** | 5 | User |

Table : ICS/SCADA System IOCs

4.1.4 Data Normalization:

Using normalization Java batch script, the text/string data in raw and aggregated data are transformed into their meaningful integer equivalent. For example, **protocol**, the ‘udp’ protocol is denoted by numerical value 1 and ‘tcp’ protocol to 2. Likewise, all other data field values, including date and time, are converted to discrete values.

4.1.5 Data Synthetization with Abnormal Samples:

To mimic abnormal behavior, collected IOCs are randomly seeded into normalized data. The IOCs are classified as System, Threat, CVE, and User attacks. To differentiate normal and abnormal samples, use the numerical notation: 1-Normal; 2-System category attack; 3-CVE category attack; 4- Threat category attack; 5 – User category attack. This is shown in table 1. The second column in Table 3 helps to increase the data entropy (a measure of randomness) because it emphasizes the benign possibility of IOC. For example, an adjustment in firewall port exemption may not in the least be an abnormal event, however, could be a short-term fix (Francia, 2017).

4.1.6 Enhancing Data Entropy**:**

Randomness is the key aspect to make dataset as realistic as possible, i.e. enhancing the data entropy without artificially seeding the data with irrational data field values. A case of such is data exfiltration activity like a sudden increase in network traffic. But, this action can also be viewed as a normal operation when an engineer might send a huge data files of network packet captures to a third party for digital analysis. So, we added some sort of entropy to the dataset by randomly marking those IOCs that may potentially be benign as either “normal “or “attack” (Francia, 2017).

4.1.7 Test Data Set:

The collected raw data contains more than 50,000 records with over 100 attributes. We cleaned and reduced this data set to 5000 records with 61 attributes and then randomly seeded 280 samples of abnormal behavior. The final dataset contains 5280 records in CSV (Comma Separated Value) file, which is utilized as an input file for the machine learning system.

4.2 Machine Learning Process:

4.2.1 Classification: To classify attacks from dataset of 5280 records, we used 5 supervised machine learning algorithms, namely Boosted Tree Ensemble (1), Complex Decision Trees (2), Weighted K-Nearest Neighbor (KNN) (3), Quadratic Support Vector Machine (SVM) (4), Scaled Conjugate Backpropagation Neural Network (5) algorithms. For algorithms 1 to 4, used MATLAB Statistics and Machine Learning Toolbox and classified data set of 5280 records with 62 features: 61 predictors and 1 response variables. We also applied 10-fold cross-validation to the data set, i.e. the data set is split into 10 folds to overcome bias (underfitting) and variance (overfitting) at the same time estimated the accuracy of each fold. For algorithm 5, we used Neural Network toolbox to recognize patterns for that given data set as input matrix and output classifier dataset as target matrix. To avoid bias and variance, we partitioned the input data set into training (60%), validation (20%), and testing (20%) datasets. The confusion matrix of algorithms 1 to 4 are shown in figure 6 and algorithm 5 confusion matrix is shown in figure 7. The 5 model’s classification results are shown in table 3.

|  |  |
| --- | --- |
| Screen Clipping | Screen Clipping |
| Boosted Tree Ensemble | Complex Decision Tree |
| Screen Clipping | Screen Clipping |
| Weighted KNN | Quadratic SVM |

Figure 6: Algorithms 1 to 4 Confusion Matrix

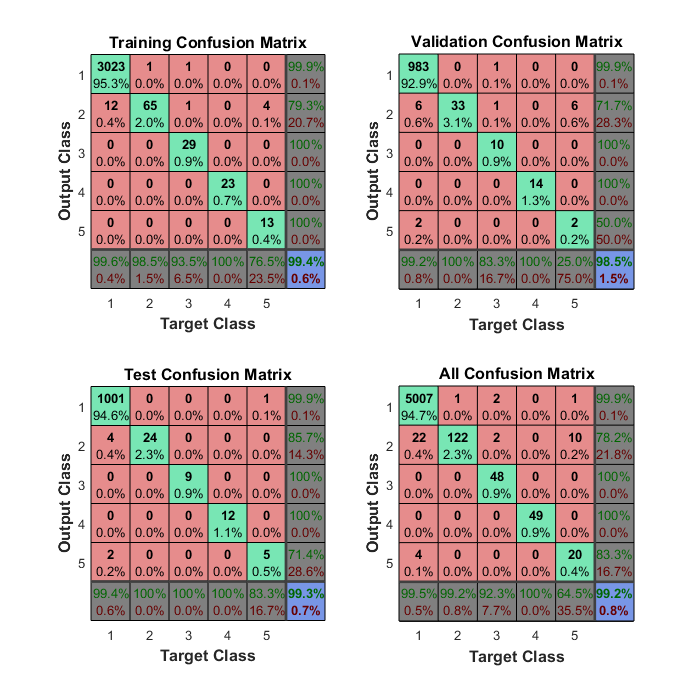


Figure : Algorithm 5 neural network confusion matrix

4.2.2 Evaluation of Outcomes and Observations:

We observed all trained algorithm models and their confusion matrices particularly compared the classification and/or pattern recognition models on 10 unique features: accuracy, training time, the number of samples correctly identified as abnormal (TP), the number of samples correctly identified as normal (TN), the number of samples incorrectly identified as abnormal (FP), the number of samples incorrectly identified as normal (FN), recall, precision, specificity, and informedness. To compute last 4 features, we used confusion matrix data values and their formulas are given below:

The reason to select TP for identifying abnormal events and TN for identifying normal events is recognizing abnormalities or attack patterns from normal events (Francia, 2017). For that, we assume an abnormal event as positive and a normal event as negative.

In this evaluation, **recall** and **precision** depict the complete discovery of abnormal events and quality in their classification respectively. In essence, the **high recall** implies the majority of the abnormal events have been detected; a **high precision** implies classification on abnormal events is accurate with the lowest misclassification. The feature **Specificity** defines the measure of distinguishing normalcy. So, a **high Specificity** shows a high rate of classifying normal events. Lastly, **informedness** feature signifies scale of how informed a predictor is for a particular condition (Francia, 2017). The comparison of five algorithms with these features is shown in figure 8.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Classifier Model** | **Accuracy (%)** | **Training Time (sec)** | **TP** | **TN** | **FP** | **FN** | **Recall (%)** | **Precision (%)** | **Specificity (%)** | **Informedness (%)** |
| ***Boosted Trees Ensemble*** | 99.20 | 7.64 | 243 | 5005 | 6 | 28 | 89.67 | 97.59 | 99.88 | 89.55 |
| ***Complex Decision Trees*** | 99.1 | 1.33 | 234 | 5007 | 16 | 26 | 90.00 | 93.60 | 99.68 | 89.68 |
| ***Weighted KNN*** | 99.2 | 2.18 | 237 | 5007 | 11 | 26 | 90.11 | 95.56 | 99.78 | 89.89 |
| ***Quadratic Support Vector Machine (SVM)*** | 99.1 | 6.33 | 233 | 5007 | 11 | 26 | 89.96 | 95.49 | 99.78 | 89.74 |
| ***Conjugate gradient backpropagation*** | 99.2 | - | 239 | 5007 | 26 | 4 | 98.35 | 90.19 | 99.48 | 97.83 |

Table : Data Classification Results

Figure : Performance Comparison of Classifiers

4.3 Visual System Analytics:

On a daily basis, the machines in ICS system generate a lot of data which can be useful or useless. As a consequence, identifying new patterns and understanding systems’ behavior will be very difficult when performing manual analysis. This is a very time consuming and tedious task. In this particular case, data visualization can be very handy. It allows access to large amounts of data in easily digestible visuals. Data visualization is the pictorial representation of data. It allows decision makers/analysts to grasp difficult system concepts and identify new patterns in existing data. For example, locating and counting destinations for outbound traffic from ICS system visually helps decision makers to identify anonymous destinations. We developed a proof of concept tool to visualize ICS system generated data using a Tableau Server and created some useful visualizations. In what follows, various visualizations of ICS security monitoring are described.

4.3.1 Monitoring Available Space Checks in Disk Drive Type. Over the time, checking of available space on each drive type is depicted in figure 9.

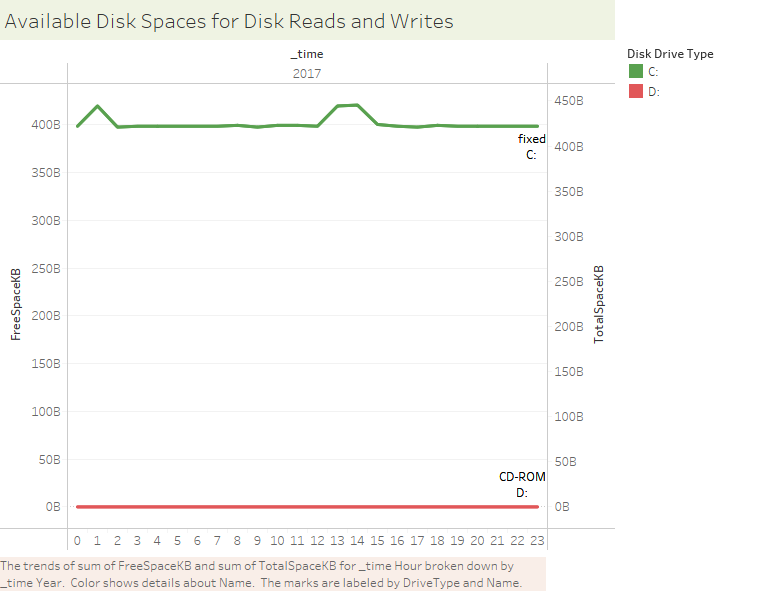


Figure : Monitoring available space checks in Disk Drive Type

The line graph represents the total space and available free space of disk drives: C & D and their drive type (fixed, CD-ROM) over the time. Precisely, the C drive has enough space, which may allow us to overlook on other several actions such as disk writes, disk reads, and disk transfers. In a security stand point, this visual system can be very useful in monitoring illegitimate activities on physical drives such as data infiltration and exfiltration.

4.3.2 Monitoring More Frequent Instances of Processes on Host Machine. It is very important to know what process instances are running because there could be a chance of misuse of system resources. The column graph, shown in figure 10 list out the top 25 distinct process running on hosts as well as more frequently running process instances.

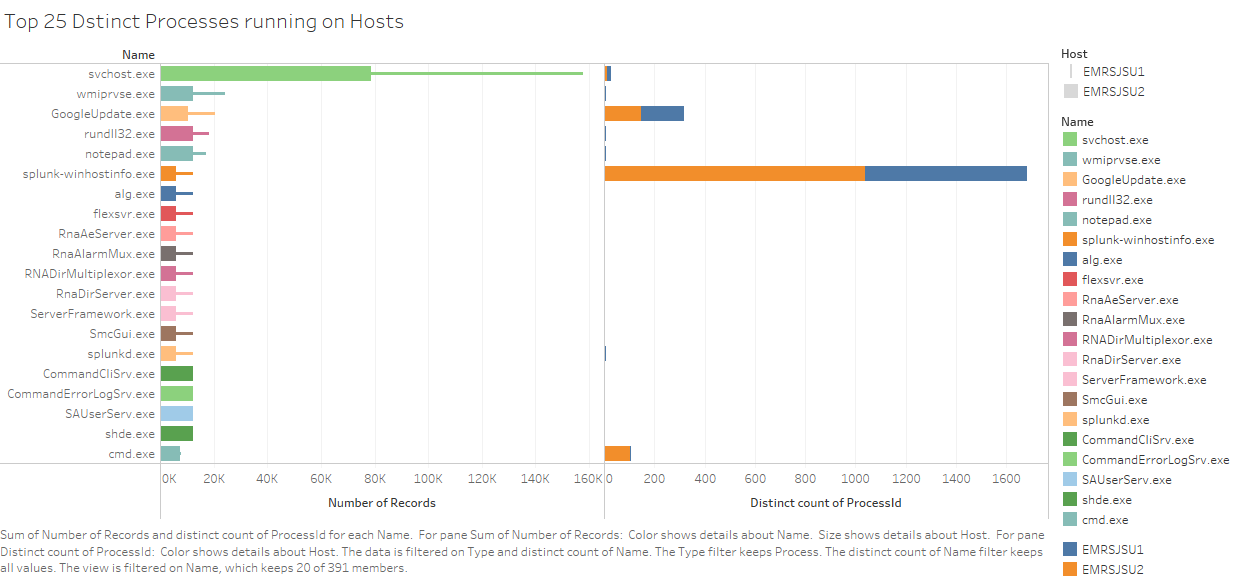


Figure : Distinct processes and instances of processes on host

Over the time, only Splunk Server, command prompt, Google update, service host, Notepad applications have shown more than instance.

4.3.3 Classifying Generated Events by Type of User. The machines in an ICS system generate predefined event codes automatically when a user performs an action on the system. The visualization in figure 11 lists out the event codes created by the type of user.

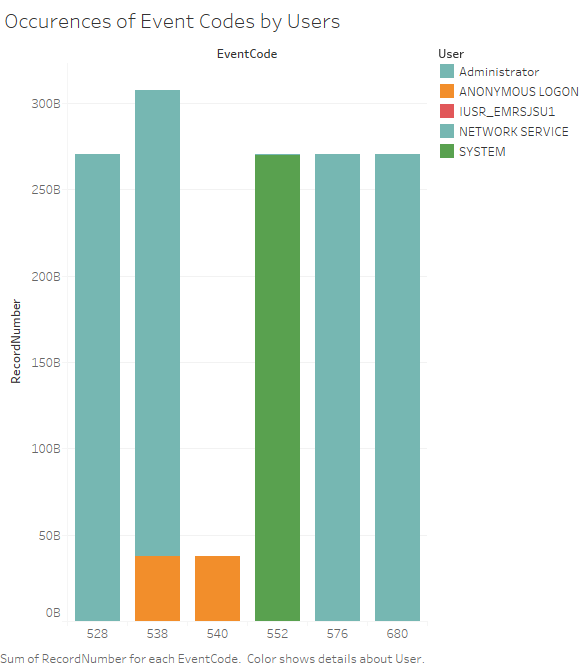


Figure : User created event code separation

The codes 528, 538, 540, 552, 576, 680 are the major event codes and each code means successful logon, user logoff, successful network logon, logon attempt using explicit credentials, special privileges assigned to user logon, and successful account logon respectively. This visualization can be very useful for the security personnel in determining possible system breaches and other illegal system activities.

4.3.4 Monitoring Event Logs Created by Users. In addition to the previous visualizations, visualizing generated event logs can be very useful in understanding the type of user and their activities. Also, this can help in checking anonymous activity and misuse of user privileges.

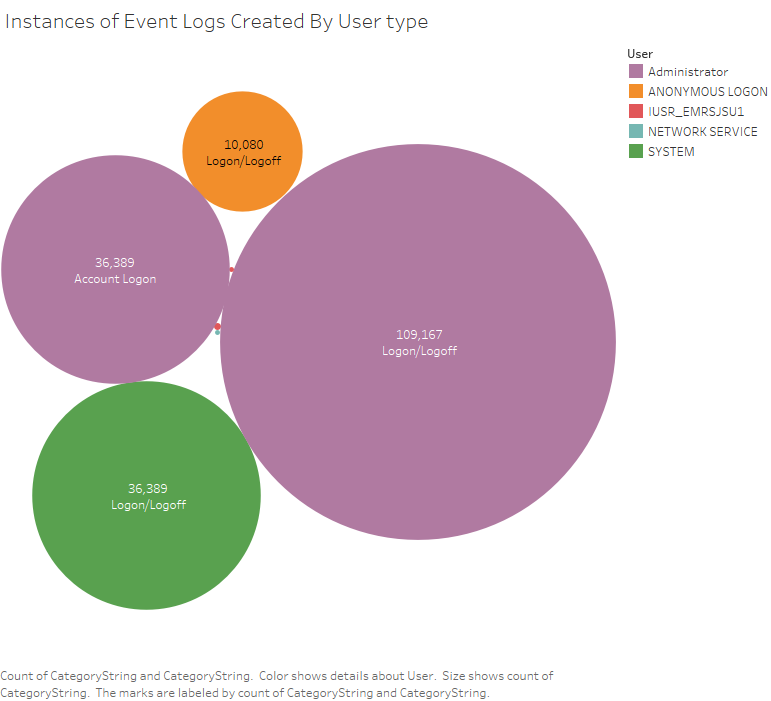


Figure : Visualizing number of logins/logouts

The bubble graph in figure 12 shows us instances and type of user activity (logon/logoff/account logon) presented by five different users, namely Administrator, System, Network Service, Anonymous logon, and User\_IMRSJSU. Interestingly, more than 10,000 anonymous logon/logoff actions have occurred along with administrator and system user logins. Further action needs to be taken to determine whether those are suspicious or legitimate.

4.3.5 Analyzing Outbound Traffic of Client IPs. It is very important to clear traffic in the network and to assign priorities for traffic request by examining current outgoing traffic created by the client IPs including protocol, used application, destination port, and destination location. The treemap in figure 13 represents outbound traffic main parameters used by three standard client IPs. On the security standpoint, the treemap provides a way of determining the amount of outgoing network traffic and facilitates the identification of data exfiltration.

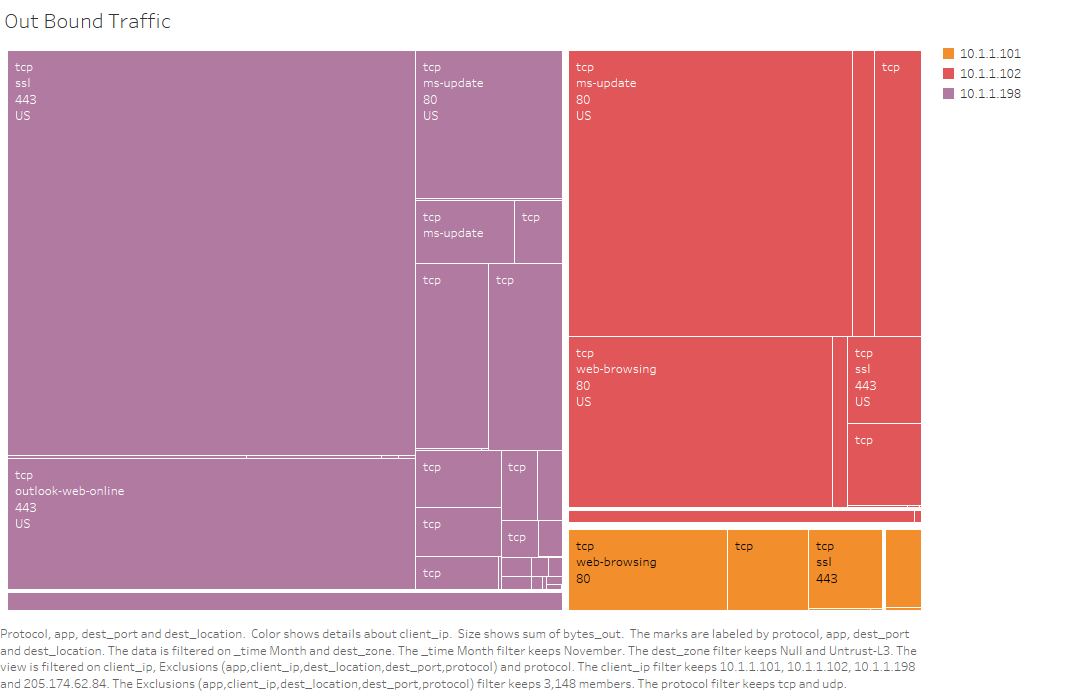


Figure : Client IPs Outbound traffic

4.3.6 Monitoring Destination IP Locations Geographically. In addition to outbound traffic, we also examined destination IP locations across the globe.

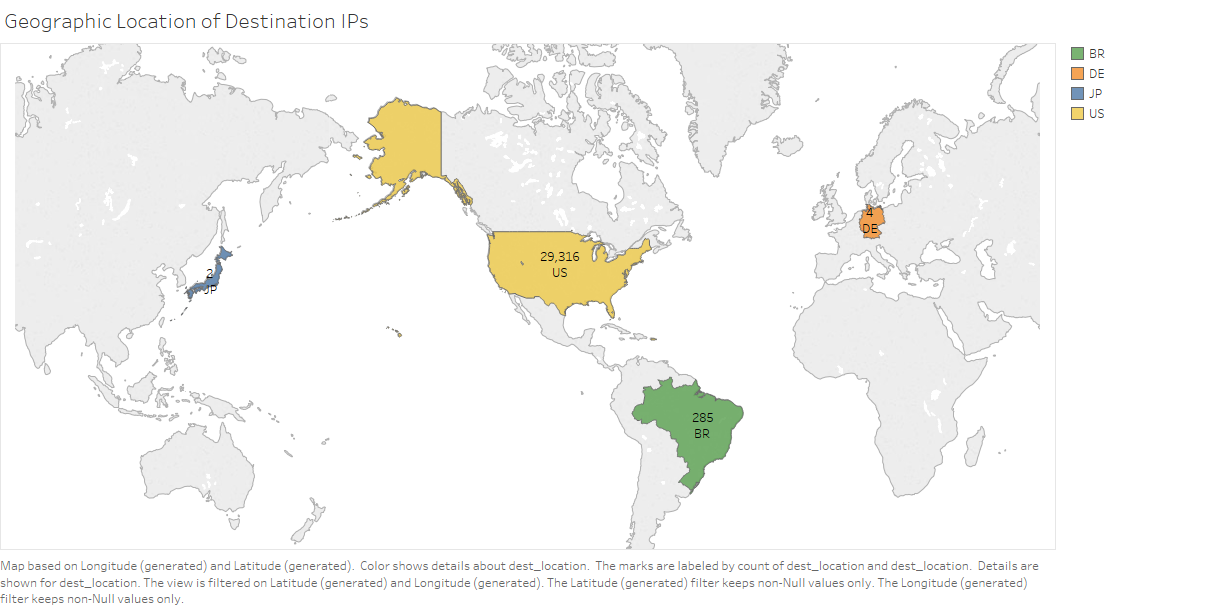


Figure : Geographic locations of destinations IP

The US, Japan, Brazil, and Denmark are the major locations that are receiving the traffic from the network and it is shown in figure 14. The significance of this visualization is in its capacity to identify the destination countries of the data being moved. Countries such as China, Russia, and other hostile nations may raise some suspicion.

4.3.7 Observation of Bandwidth Usage. Monitoring network bandwidth helps us to understand how much data flows across the network. By drawing a line graph, as shown in figure 15, one can figure out how much network capacity and internet bandwidth is required by an organization. Also, it provides help in identifying issues with Internet connectivity and in improving overall network performance.

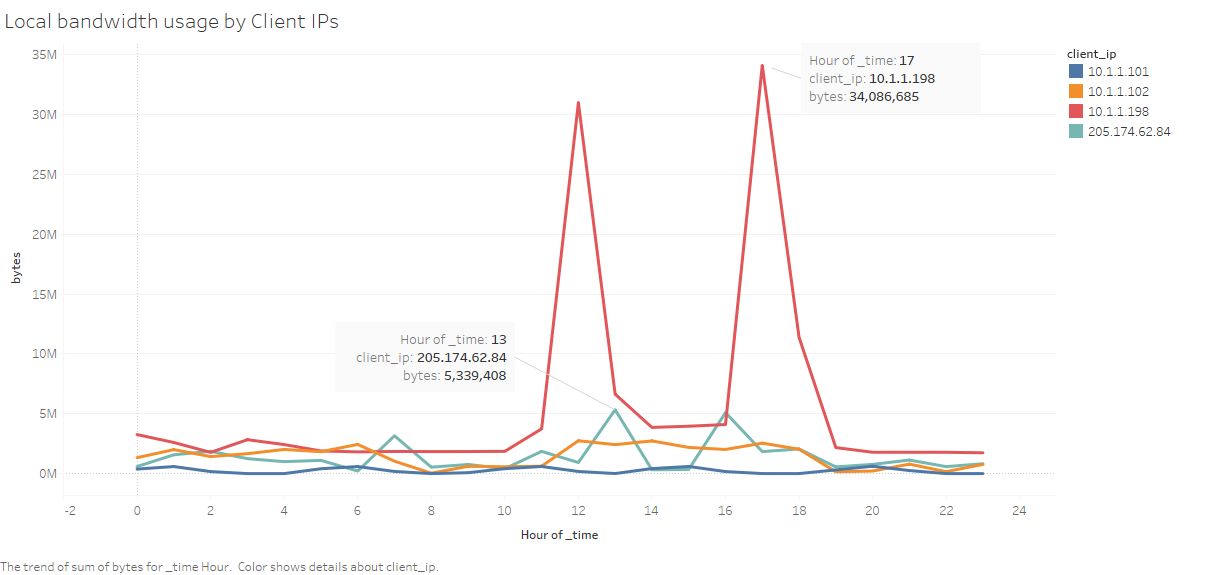


Figure : Network bandwidth usage

As shown in the graph, all hosts, except client IP 10.1.1.198, use up to 5MB per second. The client IP 10.1.1.198 is using a maximum of up to 34MB per second over that same period of the time.

4.3.8 Observing Windows Performance Monitor. The Windows performance evaluation typically measures how many bytes sent and/or received through the network adapters. The following area graph, shown in figure 16, represents the network traffic flow in megabytes.

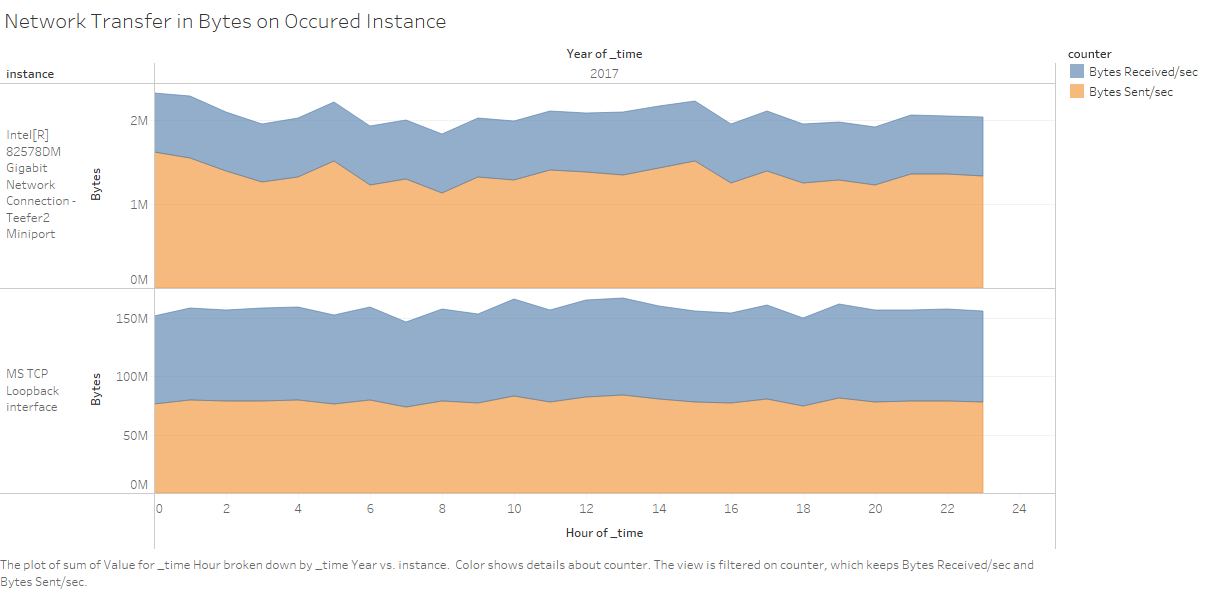


Figure : Network Transfers in Bytes

5. Conclusion and Future Enhancements:

In this research, to fulfill the great need for ICS security test datasets, we presented a strategy to collect data from an ICS system as a proof of concept framework using a Splunk Server. In addition, we collected and aggregated operational threat intelligence in real time. The test dataset is prepared by including added intelligence using Indicators of Compromise. Then we performed its evaluation for suitability to machine learning. Also, with the proof of concept tool, we created useful visual solutions from the test dataset using Tableau to enable visual analytics. The results clearly indicate that a continuous monitoring security tool for ICS system can be designed and implemented using a collection of tools such as Splunk for data collection and aggregation, MATLAB for machine intelligence, and Tableau for visual analytics. We hope that this small contribution would stimulate further research in the continuous security monitoring of ICS and SCADA systems.

With regard to future enhancement of this research project, we believe that we can further reduce the dimensionality of the test dataset using feature selection and Principal Component Analysis (PCA). Another future direction is to expand the dataset by adding extra data points related to user, system, and network behavior. In addition, expand the research towards intelligent dataset by gathering more Indicators of Compromise from various threat intelligence sources. Finally, we envision a system that can provide an automated and real-time alert system that will work in conjunction with the machine intelligent backend and the visual analytics subsystem.

# 6. References

Andress, J. (2015). Working with Indicators of Compromise. *ISSA- Developing and Connecting Cyber Security Leaders Globally*.

Andrew, N. (n.d.). *CS229 Machine Learning Course Materials.* Retrieved from CS229-Machine Learning: http://cs229.stanford.edu/materials.html

Assante, M. J., & Lee, R. M. (2015, October). The Industrial Control System Cyber Kill Chain. *SANS Institute InfoSec Reading Room*.

Case, S. I. (2016, March 18). Analysis of the Cyber Attack on the Ukrainian Power Grid. *SANS ICS and E-ISAC*.

Chen, L.-d., Sakaguchi, T., & Frolick, M. N. (2000). Data Minig Methods, Applications, and Tools. *Information Systems Management*, 65-70.

Chismon, D., & Ruks, M. (2015). *Threat Intelligence: Collecting, Analysing, Evaluating.* Retrieved from mwrinfosecurity: https://www.mwrinfosecurity.com/assets/Whitepapers/Threat-Intelligence-Whitepaper.pdf

Francia, G. A. (2017). A Machine Learning Test Data Set for Continuous Security Monitoring of Industrial Control Systems.

Jaime, B. (2015, July 28). *Open Threat Exchange (OTX) - Now Including Social Sharing of Threat Data*. Retrieved from Alien Vault: https://www.alienvault.com/blogs/security-essentials/open-threat-exchange-20-social-sharing-ups-the-value

Josh, R. (2015, June 25). *Understanding the Threat Landscape: Indicators of Compromise (IOCs)*. Retrieved from circleid internet infrastructure: http://www.circleid.com/posts/20150625\_understanding\_the\_threat\_landscape\_indicators\_of\_compromise\_iocs/

Kyle, W., & Jim, G. (2014, October 16). *Sandworm to Blacken: The SCADA Connection*. Retrieved from trend micro: http://blog.trendmicro.com/trendlabs-security-intelligence/sandworm-to-blacken-the-scada-connection/

Martin, L. (2015). *Seven Ways to Apply the Cyber Kill Chain® With A Threat Intelligence Platform.* Retrieved from lockheedmartin: http://lockheedmartin.com/content/dam/lockheed/data/corporate/documents/Seven\_Ways\_to\_Apply\_the\_Cyber\_Kill\_Chain\_with\_a\_Threat\_Intelligence\_Platform.PDF

networks, P. (2017). *AutoFocus™ New Features Guide*. Retrieved from paloalto networks: https://www.paloaltonetworks.com/content/dam/pan/en\_US/assets/pdf/framemaker/autofocus/autofocus/new-feature-guide/section\_1.pdf

Nigam, R. (2016, April 05). *SCADA Security Report 2016*. Retrieved from fortinet: https://blog.fortinet.com/2016/04/05/scada-security-report-2016

Patricia, C. (2017, January 9). *Tech Docs: Collect and Refine Threat Intelligence with MineMeld*. Retrieved from paloalto networks: http://researchcenter.paloaltonetworks.com/2017/01/tech-docs-collect-refine-threat-intelligence-minemeld/

Rouse, M. (2005, September). *SCADA (supervisory control and data acquisition)*. Retrieved from whatis.com: http://whatis.techtarget.com/definition/SCADA-supervisory-control-and-data-acquisition

Rouse, M. (2016, March). *industrial control system (ICS)*. Retrieved from whatis.com: http://whatis.techtarget.com/definition/industrial-control-system-ICS

Scott, S. (2016, August 1). *Introducing MineMeld: Simplified, Open-source Threat Intelligence Sharing*. Retrieved from paloalto networks: http://researchcenter.paloaltonetworks.com/2016/08/introducing-minemeld-simplified-open-source-threat-intelligence-sharing/

Team, C. (2015). *Cyber Threat Indicator-CISA-2015.* Retrieved from pillsburylaw: http://www.pillsburylaw.com/siteFiles/HiddenPages/Definitions.pdf

ThreatConnectTeam. (2015). *Maturing A Threat Intelligence Program.* Retrieved from ThreatConnect: https://www.threatconnect.com/wp-content/uploads/ThreatConnect-Maturing-Program-Whitepaper.pdf

ThreatConnectTeam. (2015). *Threat Intelligence Platforms.* Retrieved from ThreatConnect: http://cdn2.hubspot.net/hubfs/454298/ebook/Threat-Intel-Platform-ebook-ThreatConnect.pdf

# Appendix A – Data Collector Indexes for Splunk Input.conf file

[script://$SPLUNK\_HOME\bin\scripts\splunk-wmi.path]

disabled = 0

[WinEventLog://Security]

interval = 300

index = SP17\_eventmon

disabled = 0

[monitor://C:\TEMP\security.txt]

interval = 300

index = SP17\_SEP-security

sourcetype = security

followTail = 1

disabled = 0

[monitor://"C:\Documents and Settings\Administrator\Desktop\ICS\_reader\parsedfloat.csv"]

interval = 300

index = SP17\_floatstats

followTail = 1

disabled = 0

[monitor://C:\TEMP\system.txt]

interval = 300

index = SP17\_SEP-system

followTail = 1

disabled = 1

[monitor://C:\TEMP\packet.txt]

interval = 300

index = SP17\_SEP-packet

followTail = 1

disabled = 0

[WinHostMon://process]

interval = 300

type = process

index = SP17\_WinHostMon

disabled = 0

[WinHostMon://disk]

interval = 300

type = disk

index = SP17\_WinHostMon

disabled = 0

[WinHostMon://service]

interval = 300

type = service

index = SP17\_WinHostMon

disabled = 0

[WinRegMon://\*]

proc=DerivedTags.exe|DisplayClient.exe|DatalogServ.exe|FTA\*.exe|NMSPHost.exe|RSLinxNG.exe|RnAeServer.exe|Tagsrv.exe|RsvcHost.exe|RSVIEW\*.exe|ViewPoint.Server\*.exe|RNADirMultiplexor.exe

interval = 300

index = SP17\_regmon

baseline = 1

disabled = 0

[perfmon://Network]

index = SP17\_perfmon

interval = 300

object = Network Interface

counters = Bytes Received/sec; Bytes Sent/sec; Bytes Total/sec

instances = \*

disabled = 0

### Below does not work on Windows Server 2003 and below ###

[WinNetMon://inbound]

index = SP17\_winnetmon

direction = inbound

interval = 300

disabled = 1

[WinNetMon://outbound]

index = SP17\_winnetmon

direction = outbound

interval = 300

disabled = 1

# Appendix B – Test Data Features

|  |  |
| --- | --- |
| Attribute Name | Description |
| Process Name | The name of the running process |
| Path | The complete pathname of the running process |
| Process ID | The process ID |
| Process Time | The time the process started |
| App | The application name |
| App\_able | Status whether the application is capable of file transfer (yes/no) |
| App\_category | Application category (networking, business, general) |
| App\_default\_port | The default port used by the application |
| App\_evasive | Status whether the application is evasive from Intrusion Detection Systems (IDS) (yes/no) |
| App\_excessive\_bandwidth | Status whether the application uses excessive bandwidth (yes/no) |
| App:has\_known\_vulnerability | Status whether the application has known vulnerability (yes/no) |
| App:risk | Risk level of application (1-5) |
| App:subcategory | The application subcategory |
| App:used\_by\_malware | Status whether the application is known to be used by malware (yes/no) |
| Application | Intended use (ms-update,dns, ssl, netbios, etc) |
| Bytes\_in | Amount of incoming traffic (bytes) |
| Bytes\_out | amount of outgoing traffic (bytes) |
| Dest\_port | Network traffic destination port |
| Protocol | Network traffic protocol in use |
| Src\_IP | The source Internet Protocol (IP) address |
| Threat\_category | Threat category as reported by Palo Alto Network (PAN) traffic (malicious, benign) |
| Threat\_CVE | CVE number of threat |
| Threat\_severity | Severity of threat (Critical, High, Medium, Low) |
| Threat\_ID | Unique identifier for the threat (scan detection: 8000-8099; flood detection: 8500-8599; URL Filtering: 9999: spyware download: 20000-29999; Vulnerability exploit detection: 30000-44999; Filetype detection: 52000-52999; Data filtering detection: 60000-69999; Virus detection: 100000-2999999) |
| Threat\_name | Name of threat |
| Transport | Network protocol used |
| Type | Type of traffic payload (Threat, Traffic, etc) |
| URL | Uniform Resource Locator |
| Drive\_type | Type of storage device (fixed, removable) |
| FileSystemType | Type of file system in use (NTFS, ext4, HFS) |
| FreeSpaceKB | Free space available for use in Kbytes |
| DriveName | Label of drive |
| UsedSpaceKB | Total space used in Kbytes |
| EventCode | Windows security event code |
| Logon\_ID | User ID |
| User | User name |
| Event\_time | Time of event |
| Network\_transfer | Amount of network transfer in Kbytes |
| Network\_time | Time of network capture |
| Network\_transfer | Amount of network transfer in Kbytes |
| Traffic\_type | Network traffic type (send/receive) |
| Reg\_time | Time of registry activity |
| Rdata | Registry data |
| Revent\_status | Registry event status (successful/unsuccessful) |
| Key\_path | Full path to registry item |
| Rpid | Process ID of registry event |
| Reg\_time | Time of registry activity |
| Rprocess\_image | Name of registry process |
| Registry\_type | Registry type (CreateKey, ChangeKey, DeleteKey) |
| SEP\_time | Symantec Endpoint Process (SEP) time |
| SEP\_action | SEP action (blocked, allowed) |
| SEP\_app | Application monitored by SEP |
| Direction | Direction of traffic monitored by SEP (incoming/outgoing) |
| SEP\_local\_port | Local port monitored by SEP |
| SEP\_packet\_ID | Application ID monitored by SEP |
| SEP\_remote\_host | Remote host name monitored by SEP |
| SEP\_tag | SEP tag name |
| Serv\_Desc | Service Description |
| Serv\_Name | Name of service |
| StartMode | Service start mode (Auto/Manual) |
| Serv\_Start | Status of service start (TRUE/FALSE) |
| Serv\_time | Time service started |
| Record\_type | Flag for record type (1-Normal, 2-System, 3-CVE, 4-Threat, 5-User) |

# Appendix C – Python Script for IOC Collection

**OTXReceiver\_ICS\_iocs.py**:

from OTXv2 import OTXv2, IndicatorTypes

import re

import os

import sys

import traceback

import argparse

HASH\_BLACKLIST = [ 'e617348b8947f28e2a280dd93c75a6ad', '125da188e26bd119ce8cad7eeb1fc2dfa147ad47',

'06f7826c2862d184a49e3672c0aa6097b11e7771a4bf613ec37941236c1a8e20' ]

tagslist = ['energy', 'blackenergy', 'scada', 'industrial', 'power plant', 'plant', 'nuclear', 'defense',

'manufacturing', 'cyber', 'automation', 'power grid', 'ics', 'military', 'palo alto', 'Energy','paloalto']

def my\_escape(string):

return re.sub(r'([\-\(\)\.\[\]\{\}\\\+])',r'\\\1',string)

class OTXReceiver:

# IOC strings

hash\_iocs = ""

filename\_iocs = ""

c2\_iocs = ""

#output format

separator = ';'

use\_csv\_header = False

extension = 'txt'

hash\_upper = False

filename\_regex\_out = True

# event container

events =[]

#debug, proxy

def get\_ics\_pulses(self, api\_key, siem\_mode):

#self.debug = debug

self.otx = OTXv2(api\_key)

if siem\_mode:

self.separator = ","

self.use\_csv\_header = True

self.extension = "csv"

self.hash\_upper = True

self.filename\_regex\_out = False

def get\_iocs\_last(self):

# mtime = (datetime.now() - timedelta(days=days\_to\_load)).isoformat()

try:

print("Starting OTX feed download ...")

pulses = self.otx.getall()

if len(pulses) > 0:

for pulse in pulses:

pulsetags = pulse['tags']

for tag in pulsetags:

if tag in tagslist:

self.events.append(pulse)

print("Download complete - %s events received" % len(self.events))

else:

print("Download failed - no events received (check your Internet connection)")

# json\_normalize(self.events)

except Exception as e:

traceback.print\_exc()

def write\_iocs(self, ioc\_folder):

hash\_ioc\_file = os.path.join(ioc\_folder, "otx-hash-iocs.{0}".format(self.extension))

filename\_ioc\_file = os.path.join(ioc\_folder, "otx-filename-iocs.{0}".format(self.extension))

c2\_ioc\_file = os.path.join(ioc\_folder, "otx-c2-iocs.{0}".format(self.extension))

print("Processing indicators ...")

for event in self.events:

try:

for indicator in event["indicators"]:

if indicator["type"] in ('FileHash-MD5', 'FileHash-SHA1', 'FileHash-SHA256') and \

indicator["indicator"] not in HASH\_BLACKLIST:

hash = indicator["indicator"]

if self.hash\_upper:

hash = indicator["indicator"].upper()

self.hash\_iocs += "{0}{3}{1} {2}\n".format(

hash,

event["name"].encode('unicode-escape'),

" / ".join(event["references"])[:80],

self.separator)

if indicator["type"] == 'FilePath':

filename = indicator["indicator"]

if self.filename\_regex\_out:

filename = my\_escape(indicator["indicator"])

self.filename\_iocs += "{0}{3}{1} {2}\n".format(

filename,

event["name"].encode('unicode-escape'),

" / ".join(event["references"])[:80],

self.separator)

if indicator["type"] in ('domain', 'hostname', 'IPv4', 'IPv6', 'CIDR'):

self.c2\_iocs += "{0}{3}{1} {2}\n".format(

indicator["indicator"],

event["name"].encode('unicode-escape'),

" / ".join(event["references"])[:80],

self.separator)

except Exception as e:

traceback.print\_exc()

# Write to files

with open(hash\_ioc\_file, "w") as hash\_fh:

if self.use\_csv\_header:

hash\_fh.write('hash{0}description\n'.format(self.separator))

hash\_fh.write(self.hash\_iocs)

print("{0} hash iocs written to {1}".format(self.hash\_iocs.count('\n'), hash\_ioc\_file))

with open(filename\_ioc\_file, "w") as fn\_fh:

if self.use\_csv\_header:

fn\_fh.write('filename{0}description\n'.format(self.separator))

fn\_fh.write(self.filename\_iocs)

print("{0} filename iocs written to {1}".format(self.filename\_iocs.count('\n'), filename\_ioc\_file))

with open(c2\_ioc\_file, "w") as c2\_fh:

if self.use\_csv\_header:

c2\_fh.write('host{0}description\n'.format(self.separator))

c2\_fh.write(self.c2\_iocs)

print("{0} c2 iocs written to {1}".format(self.c2\_iocs.count('\n'), c2\_ioc\_file))

OTX\_KEY = "926c1e1c8338ff5be230d0bac854fd5e3f0206189310c743f501d75e599e5d28"

siem ='True'

# Create a receiver

otx\_receiver = OTXReceiver()

#get ICS/SCADA pulse events

otx\_receiver.get\_ics\_pulses(OTX\_KEY, siem)

# Retrieve the events and store the ICS\_IOCs

otx\_receiver.get\_iocs\_last()

# Write ICS\_IOC files

otx\_receiver.write\_iocs('Alienvault\_OTX\_ICS\_IOCs')

# Appendix D – Sample of Normalized Test Dataset

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ProcName | Path | app | bytes\_in | bytes\_out | …. | Attack1 | Attack2 | Attack3 | Attack4 | Attack5 |
| 1 | 1 | 1 | 0 | 92 | … | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 62 | 443 | … | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 6068 | 660 | … | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 186 | 87 | … | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 66 | 390 | … | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 6068 | 660 | … | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 62 | 470 | … | 1 | 0 | 0 | 0 | 1 |

# Appendix E – Scripts for Data Classification

|  |  |  |
| --- | --- | --- |
| **Boosted Tree Ensembles Training Script** | **Complex Decision Trees Training Script** | **Weighted KNN Training Script** |
| function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % returns a trained classifier and its accuracy. This code recreates the  % classification model trained in Classification Learner app. Use the  % generated code to automate training the same model with new data, or to  % learn how to programmatically train models.  %  % Input:  % trainingData: a matrix with the same number of columns and data type  % as imported into the app.  %  % Output:  % trainedClassifier: a struct containing the trained classifier. The  % struct contains various fields with information about the trained  % classifier.  %  % trainedClassifier.predictFcn: a function to make predictions on new  % data.  %  % validationAccuracy: a double containing the accuracy in percent. In  % the app, the History list displays this overall accuracy score for  % each model.  %  % Use the code to train the model with new data. To retrain your  % classifier, call the function from the command line with your original  % data or new data as the input argument trainingData.  %  % For example, to retrain a classifier trained with the original data set  % T, enter:  % [trainedClassifier, validationAccuracy] = trainClassifier(T)  %  % To make predictions with the returned 'trainedClassifier' on new data T2,  % use  % yfit = trainedClassifier.predictFcn(T2)  %  % T2 must be a matrix containing only the predictor columns used for  % training. For details, enter:  % trainedClassifier.HowToPredict    % Auto-generated by MATLAB on 05-Nov-2017 23:09:48      % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Train a classifier  % This code specifies all the classifier options and trains the classifier.  template = templateTree(...  'MaxNumSplits', 20);  classificationEnsemble = fitcensemble(...  predictors, ...  response, ...  'Method', 'AdaBoostM2', ...  'NumLearningCycles', 30, ...  'Learners', template, ...  'LearnRate', 0.1, ...  'ClassNames', [1; 2; 3; 4; 5]);    % Create the result struct with predict function  predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);  ensemblePredictFcn = @(x) predict(classificationEnsemble, x);  trainedClassifier.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));    % Add additional fields to the result struct  trainedClassifier.ClassificationEnsemble = classificationEnsemble;  trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2017a.';  trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing ''c'' with the name of the variable that is this struct, e.g. ''trainedModel''. \n \nX must contain exactly 61 columns because this model was trained using 61 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into the app. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''), ''appclassification\_exportmodeltoworkspace'')">How to predict using an exported model</a>.');    % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Perform cross-validation  partitionedModel = crossval(trainedClassifier.ClassificationEnsemble, 'KFold', 10);    % Compute validation predictions  [validationPredictions, validationScores] = kfoldPredict(partitionedModel);    % Compute validation accuracy  validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError'); | function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % returns a trained classifier and its accuracy. This code recreates the  % classification model trained in Classification Learner app. Use the  % generated code to automate training the same model with new data, or to  % learn how to programmatically train models.  %  % Input:  % trainingData: a matrix with the same number of columns and data type  % as imported into the app.  %  % Output:  % trainedClassifier: a struct containing the trained classifier. The  % struct contains various fields with information about the trained  % classifier.  %  % trainedClassifier.predictFcn: a function to make predictions on new  % data.  %  % validationAccuracy: a double containing the accuracy in percent. In  % the app, the History list displays this overall accuracy score for  % each model.  %  % Use the code to train the model with new data. To retrain your  % classifier, call the function from the command line with your original  % data or new data as the input argument trainingData.  %  % For example, to retrain a classifier trained with the original data set  % T, enter:  % [trainedClassifier, validationAccuracy] = trainClassifier(T)  %  % To make predictions with the returned 'trainedClassifier' on new data T2,  % use  % yfit = trainedClassifier.predictFcn(T2)  %  % T2 must be a matrix containing only the predictor columns used for  % training. For details, enter:  % trainedClassifier.HowToPredict    % Auto-generated by MATLAB on 05-Nov-2017 23:47:20      % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Train a classifier  % This code specifies all the classifier options and trains the classifier.  classificationTree = fitctree(...  predictors, ...  response, ...  'SplitCriterion', 'gdi', ...  'MaxNumSplits', 100, ...  'Surrogate', 'off', ...  'ClassNames', [1; 2; 3; 4; 5]);    % Create the result struct with predict function  predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);  treePredictFcn = @(x) predict(classificationTree, x);  trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));    % Add additional fields to the result struct  trainedClassifier.ClassificationTree = classificationTree;  trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2017a.';  trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing ''c'' with the name of the variable that is this struct, e.g. ''trainedModel''. \n \nX must contain exactly 61 columns because this model was trained using 61 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into the app. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''), ''appclassification\_exportmodeltoworkspace'')">How to predict using an exported model</a>.');    % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Perform cross-validation  partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold', 10);    % Compute validation predictions  [validationPredictions, validationScores] = kfoldPredict(partitionedModel);    % Compute validation accuracy  validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError'); | function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % returns a trained classifier and its accuracy. This code recreates the  % classification model trained in Classification Learner app. Use the  % generated code to automate training the same model with new data, or to  % learn how to programmatically train models.  %  % Input:  % trainingData: a matrix with the same number of columns and data type  % as imported into the app.  %  % Output:  % trainedClassifier: a struct containing the trained classifier. The  % struct contains various fields with information about the trained  % classifier.  %  % trainedClassifier.predictFcn: a function to make predictions on new  % data.  %  % validationAccuracy: a double containing the accuracy in percent. In  % the app, the History list displays this overall accuracy score for  % each model.  %  % Use the code to train the model with new data. To retrain your  % classifier, call the function from the command line with your original  % data or new data as the input argument trainingData.  %  % For example, to retrain a classifier trained with the original data set  % T, enter:  % [trainedClassifier, validationAccuracy] = trainClassifier(T)  %  % To make predictions with the returned 'trainedClassifier' on new data T2,  % use  % yfit = trainedClassifier.predictFcn(T2)  %  % T2 must be a matrix containing only the predictor columns used for  % training. For details, enter:  % trainedClassifier.HowToPredict    % Auto-generated by MATLAB on 05-Nov-2017 23:49:09      % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Train a classifier  % This code specifies all the classifier options and trains the classifier.  classificationKNN = fitcknn(...  predictors, ...  response, ...  'Distance', 'Euclidean', ...  'Exponent', [], ...  'NumNeighbors', 10, ...  'DistanceWeight', 'SquaredInverse', ...  'Standardize', true, ...  'ClassNames', [1; 2; 3; 4; 5]);    % Create the result struct with predict function  predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);  knnPredictFcn = @(x) predict(classificationKNN, x);  trainedClassifier.predictFcn = @(x) knnPredictFcn(predictorExtractionFcn(x));    % Add additional fields to the result struct  trainedClassifier.ClassificationKNN = classificationKNN;  trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2017a.';  trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing ''c'' with the name of the variable that is this struct, e.g. ''trainedModel''. \n \nX must contain exactly 61 columns because this model was trained using 61 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into the app. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''), ''appclassification\_exportmodeltoworkspace'')">How to predict using an exported model</a>.');    % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Perform cross-validation  partitionedModel = crossval(trainedClassifier.ClassificationKNN, 'KFold', 10);    % Compute validation predictions  [validationPredictions, validationScores] = kfoldPredict(partitionedModel);    % Compute validation accuracy  validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError'); |

|  |  |
| --- | --- |
| **Quadratic SVM Training Script** | **Neural Network Training Script** |
| function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)  % returns a trained classifier and its accuracy. This code recreates the  % classification model trained in Classification Learner app. Use the  % generated code to automate training the same model with new data, or to  % learn how to programmatically train models.  %  % Input:  % trainingData: a matrix with the same number of columns and data type  % as imported into the app.  %  % Output:  % trainedClassifier: a struct containing the trained classifier. The  % struct contains various fields with information about the trained  % classifier.  %  % trainedClassifier.predictFcn: a function to make predictions on new  % data.  %  % validationAccuracy: a double containing the accuracy in percent. In  % the app, the History list displays this overall accuracy score for  % each model.  %  % Use the code to train the model with new data. To retrain your  % classifier, call the function from the command line with your original  % data or new data as the input argument trainingData.  %  % For example, to retrain a classifier trained with the original data set  % T, enter:  % [trainedClassifier, validationAccuracy] = trainClassifier(T)  %  % To make predictions with the returned 'trainedClassifier' on new data T2,  % use  % yfit = trainedClassifier.predictFcn(T2)  %  % T2 must be a matrix containing only the predictor columns used for  % training. For details, enter:  % trainedClassifier.HowToPredict    % Auto-generated by MATLAB on 05-Nov-2017 23:50:19      % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Train a classifier  % This code specifies all the classifier options and trains the classifier.  template = templateSVM(...  'KernelFunction', 'polynomial', ...  'PolynomialOrder', 2, ...  'KernelScale', 'auto', ...  'BoxConstraint', 1, ...  'Standardize', true);  classificationSVM = fitcecoc(...  predictors, ...  response, ...  'Learners', template, ...  'Coding', 'onevsone', ...  'ClassNames', [1; 2; 3; 4; 5]);    % Create the result struct with predict function  predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);  svmPredictFcn = @(x) predict(classificationSVM, x);  trainedClassifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));    % Add additional fields to the result struct  trainedClassifier.ClassificationSVM = classificationSVM;  trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2017a.';  trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing ''c'' with the name of the variable that is this struct, e.g. ''trainedModel''. \n \nX must contain exactly 61 columns because this model was trained using 61 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into the app. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''), ''appclassification\_exportmodeltoworkspace'')">How to predict using an exported model</a>.');    % Extract predictors and response  % This code processes the data into the right shape for training the  % model.  % Convert input to table  inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61', 'column\_62'});    predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5', 'column\_6', 'column\_7', 'column\_8', 'column\_9', 'column\_10', 'column\_11', 'column\_12', 'column\_13', 'column\_14', 'column\_15', 'column\_16', 'column\_17', 'column\_18', 'column\_19', 'column\_20', 'column\_21', 'column\_22', 'column\_23', 'column\_24', 'column\_25', 'column\_26', 'column\_27', 'column\_28', 'column\_29', 'column\_30', 'column\_31', 'column\_32', 'column\_33', 'column\_34', 'column\_35', 'column\_36', 'column\_37', 'column\_38', 'column\_39', 'column\_40', 'column\_41', 'column\_42', 'column\_43', 'column\_44', 'column\_45', 'column\_46', 'column\_47', 'column\_48', 'column\_49', 'column\_50', 'column\_51', 'column\_52', 'column\_53', 'column\_54', 'column\_55', 'column\_56', 'column\_57', 'column\_58', 'column\_59', 'column\_60', 'column\_61'};  predictors = inputTable(:, predictorNames);  response = inputTable.column\_62;  isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];    % Perform cross-validation  partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold', 10);    % Compute validation predictions  [validationPredictions, validationScores] = kfoldPredict(partitionedModel);    % Compute validation accuracy  validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError'); | % Solve a Pattern Recognition Problem with a Neural Network  % Script generated by Neural Pattern Recognition app  % Created 06-Nov-2017 16:48:28  %  % This script assumes these variables are defined:  %  % InputsAttack5K - input data.  % OutputsAttack5K - target data.    x = InputsAttack5K';  t = OutputsAttack5K';    % Choose a Training Function  % For a list of all training functions type: help nntrain  % 'trainlm' is usually fastest.  % 'trainbr' takes longer but may be better for challenging problems.  % 'trainscg' uses less memory. Suitable in low memory situations.  trainFcn = 'trainscg'; % Scaled conjugate gradient backpropagation.    % Create a Pattern Recognition Network  hiddenLayerSize = 10;  net = patternnet(hiddenLayerSize);    % Choose Input and Output Pre/Post-Processing Functions  % For a list of all processing functions type: help nnprocess  net.input.processFcns = {'removeconstantrows','mapminmax'};  net.output.processFcns = {'removeconstantrows','mapminmax'};    % Setup Division of Data for Training, Validation, Testing  % For a list of all data division functions type: help nndivide  net.divideFcn = 'dividerand'; % Divide data randomly  net.divideMode = 'sample'; % Divide up every sample  net.divideParam.trainRatio = 60/100;  net.divideParam.valRatio = 20/100;  net.divideParam.testRatio = 20/100;    % Choose a Performance Function  % For a list of all performance functions type: help nnperformance  net.performFcn = 'crossentropy'; % Cross-Entropy    % Choose Plot Functions  % For a list of all plot functions type: help nnplot  net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...  'plotconfusion', 'plotroc'};    % Train the Network  [net,tr] = train(net,x,t);    % Test the Network  y = net(x);  e = gsubtract(t,y);  performance = perform(net,t,y)  tind = vec2ind(t);  yind = vec2ind(y);  percentErrors = sum(tind ~= yind)/numel(tind);    % Recalculate Training, Validation and Test Performance  trainTargets = t .\* tr.trainMask{1};  valTargets = t .\* tr.valMask{1};  testTargets = t .\* tr.testMask{1};  trainPerformance = perform(net,trainTargets,y)  valPerformance = perform(net,valTargets,y)  testPerformance = perform(net,testTargets,y)    % View the Network  view(net)    % Plots  % Uncomment these lines to enable various plots.  %figure, plotperform(tr)  %figure, plottrainstate(tr)  %figure, ploterrhist(e)  %figure, plotconfusion(t,y)  %figure, plotroc(t,y)    % Deployment  % Change the (false) values to (true) to enable the following code blocks.  % See the help for each generation function for more information.  if (false)  % Generate MATLAB function for neural network for application  % deployment in MATLAB scripts or with MATLAB Compiler and Builder  % tools, or simply to examine the calculations your trained neural  % network performs.  genFunction(net,'myNeuralNetworkFunction');  y = myNeuralNetworkFunction(x);  end  if (false)  % Generate a matrix-only MATLAB function for neural network code  % generation with MATLAB Coder tools.  genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');  y = myNeuralNetworkFunction(x);  end  if (false)  % Generate a Simulink diagram for simulation or deployment with.  % Simulink Coder tools.  gensim(net);  end |