Iowa State University
College of Engineering

ADSICS

Anomaly Detection System for Industrial Control Systems

SDMay22-38 : Alex Nicolellis, Muhamed Stilic, Pallavi Santhosh, Jung Ho Suh

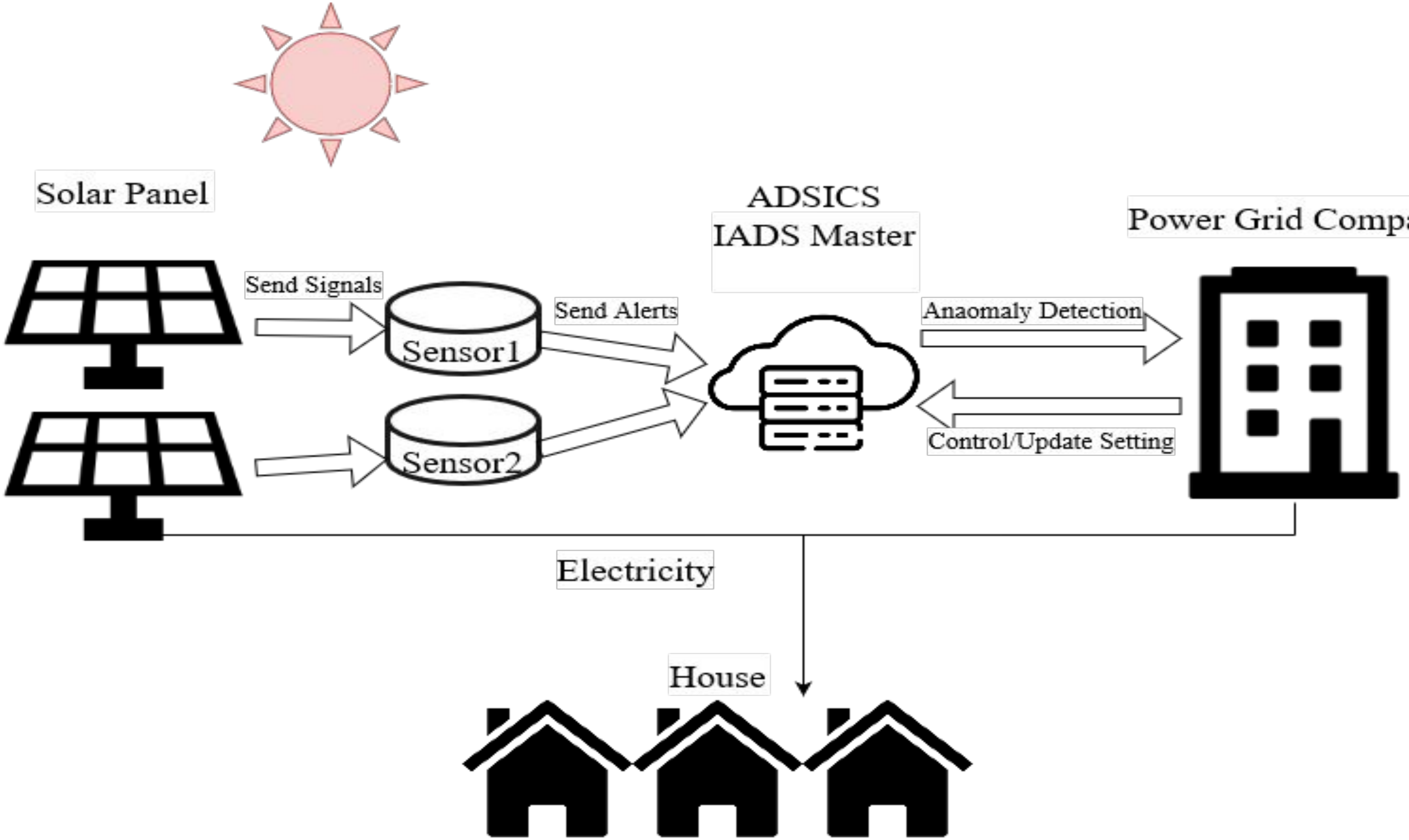
Client : Dr. Manimaran Govindarasu

Advisors : Dr. Manimaran Govindarasu, Moataz Abdelkhalek

Problem Statement

- Attacks on power distribution companies are now more common due to the increased use of IoT devices and the lack of security on power grid systems.
- ADSICS is a surveillance program that detects and prevents cyber attacks using anomaly detection.

Conceptual Sketch



Requirements & Constraints

Functional Requirements:

- Use machine learning to detect network anomalies
- Verify incoming alerts and discard false positives
- Display alerts for easy human understanding
- Present temporal and spatial details for each alert

Non-Functional Requirements:

- Alerts should be presented intuitively
- Alerts should be received within 10ms
- The system should be able to handle a large volume of alerts
- The system should be reliable and maintain uptime continuously

Constraints:

- The anomaly detection must use SecurityOnion tools (specifically Elasticsearch)

Standards

Standard	Application	Justification
IEEE 692-2013	IADS SENSOR IADS MASTER	Addresses cybersecurity and control related equipment requirements for threat assessment.
ISO IEC 27039-2015	IADS MASTER	Provides guidelines for selection, deployment, and operations of intrusion detection system detection and prevention systems.
ISO/IEC 27017:2015	CLOUD SERVER	Provides guidance on the information security aspects of cloud computing, recommending the implementation of cloud-specific information security controls that supplement the guidance of the **ISO/IEC 27002** and **ISO/IEC 27001** standards.
IEEE 1711.2-2019	IADS SENSOR	Protects communication of intelligent devices in the power industry.
IEEE 802	IADS SENSOR	Describes recommended practices for communication over various types of networks, such as wireless networks.

Market Survey

- Previous work by client - D-IDS for Cyber-Physical DER Modbus System
- Datasets: KDD, IDS, NSL+KDD, IoT23
- Anomaly detection algorithms:
    - Decision Tree
    - Random Forest
    - K-Nearest Neighbor
    - Support Vector Machine
    - Deep Neural Network

System Architecture

Functional Decomposition:

IADS Sensor, IADS Master

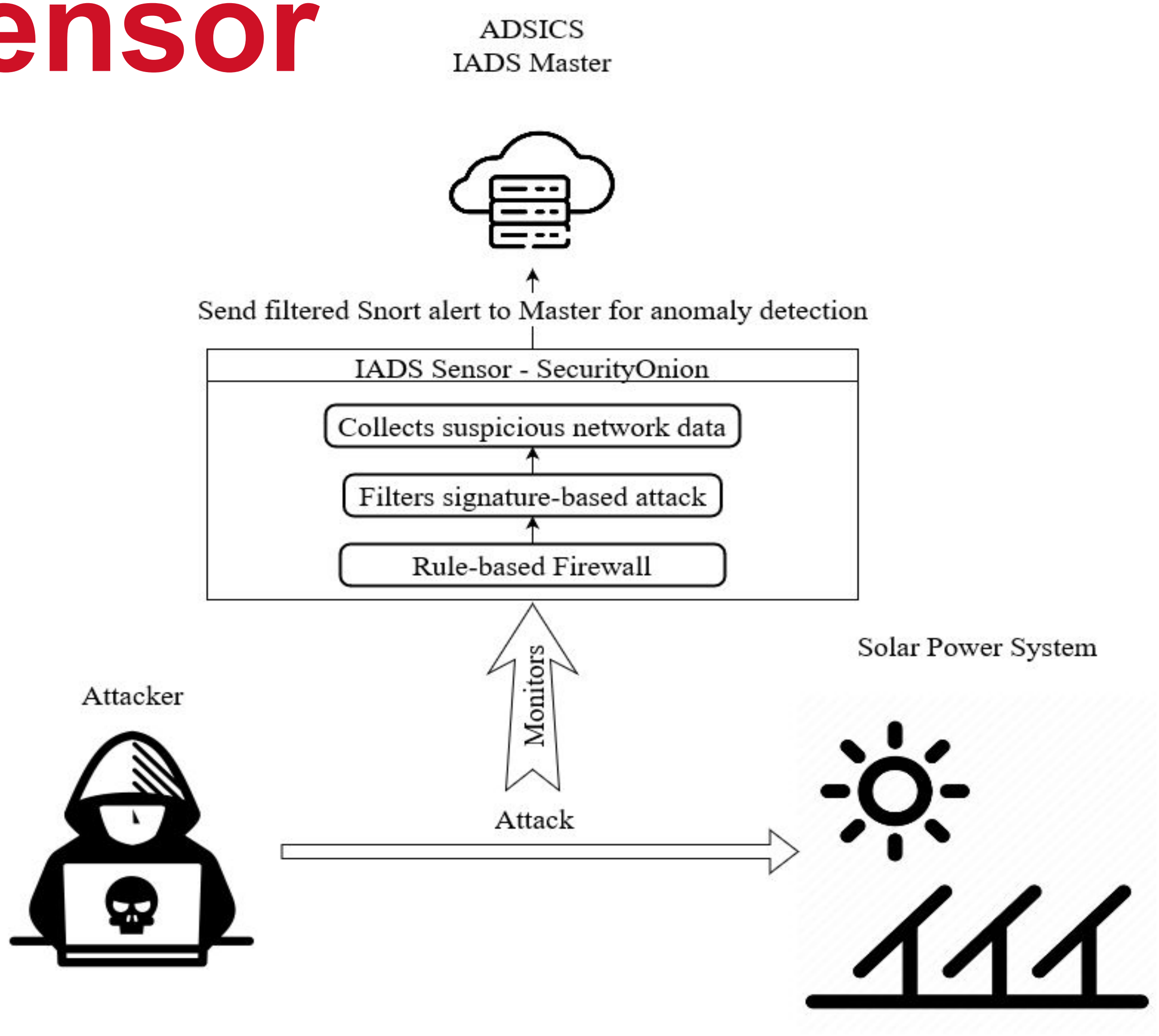
Operating System:

Kali Linux, Linux(SO), and Windows XP

Software Architecture:

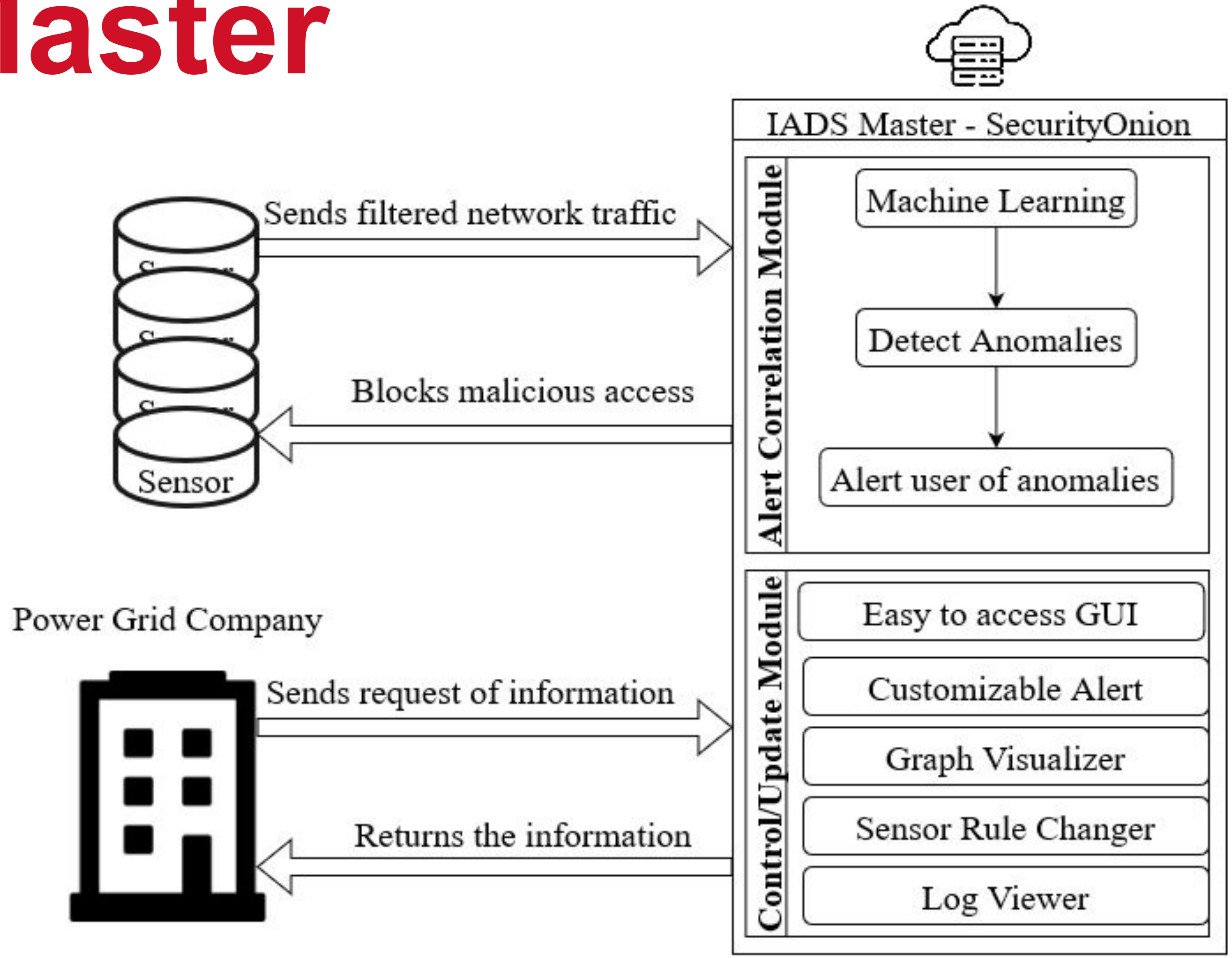
- vSphere - virtual machine
- SecurityOnion
  - ElasticSearch - machine learning
  - Kibana - visualization
  - SNORT - filtering

IADS Sensor



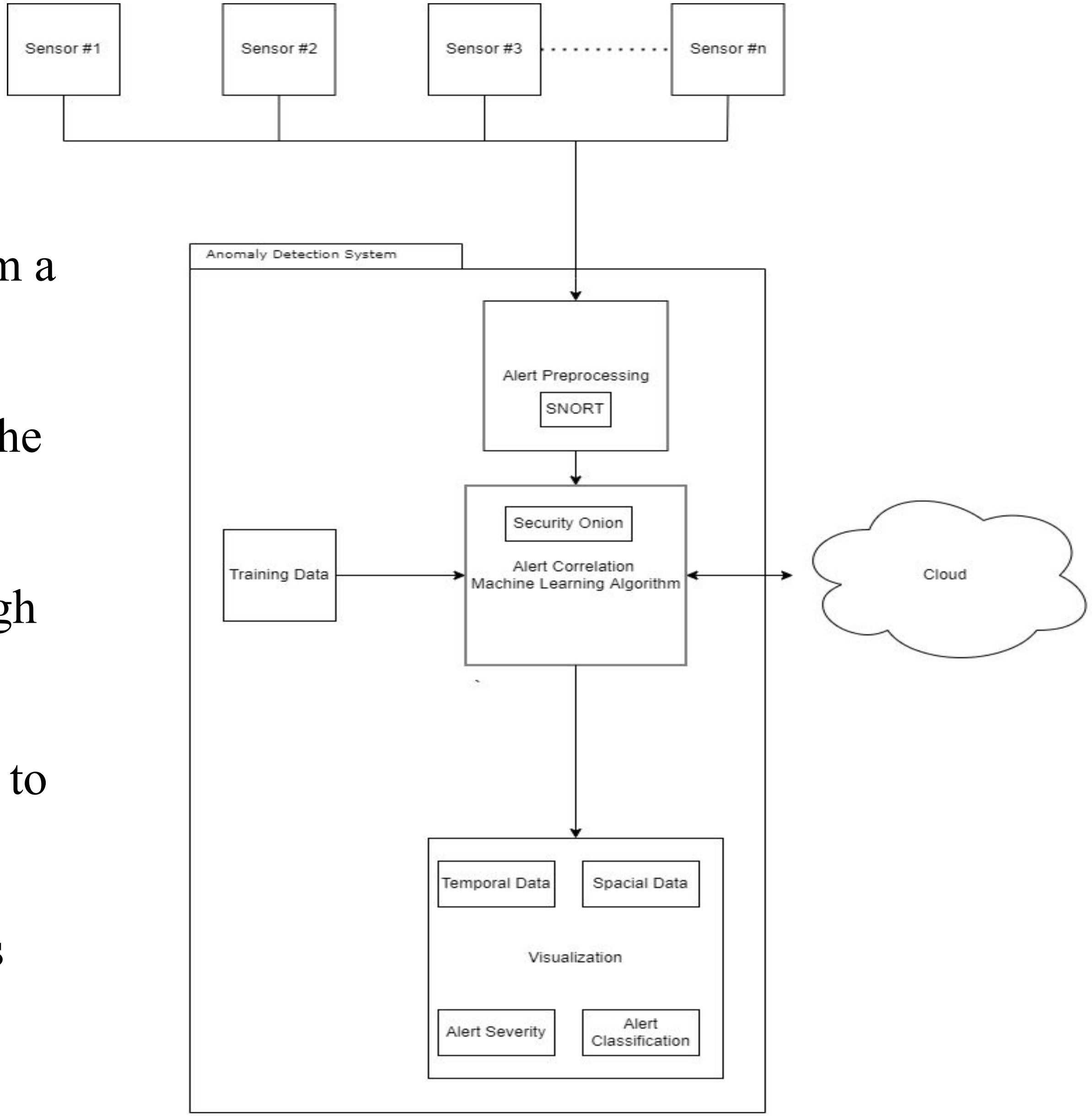
ADSICS
IADS Master
Send filtered Snort alert to Master for anomaly detection
IADS Sensor - SecurityOnion
Collects suspicious network data
Filters signature-based attack
Rule-based Firewall
Monitors
Solar Power System
Attacker
Attack

IADS Master

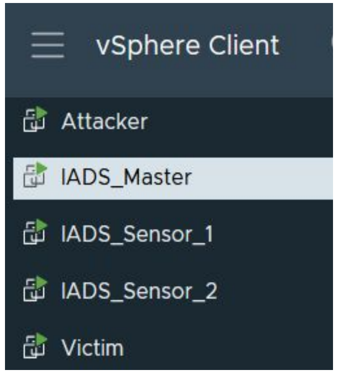

IADS Master - SecurityOnion
Alert Correlation Module
Machine Learning
Detect Anomalies
Alert user of anomalies
Control/Update Module
Easy to access GUI
Customizable Alert
Graph Visualizer
Sensor Rule Changer
Log Viewer
Sends filtered network traffic
Blocks malicious access
Sensor
Power Grid Company
Sends request of information
Returns the information

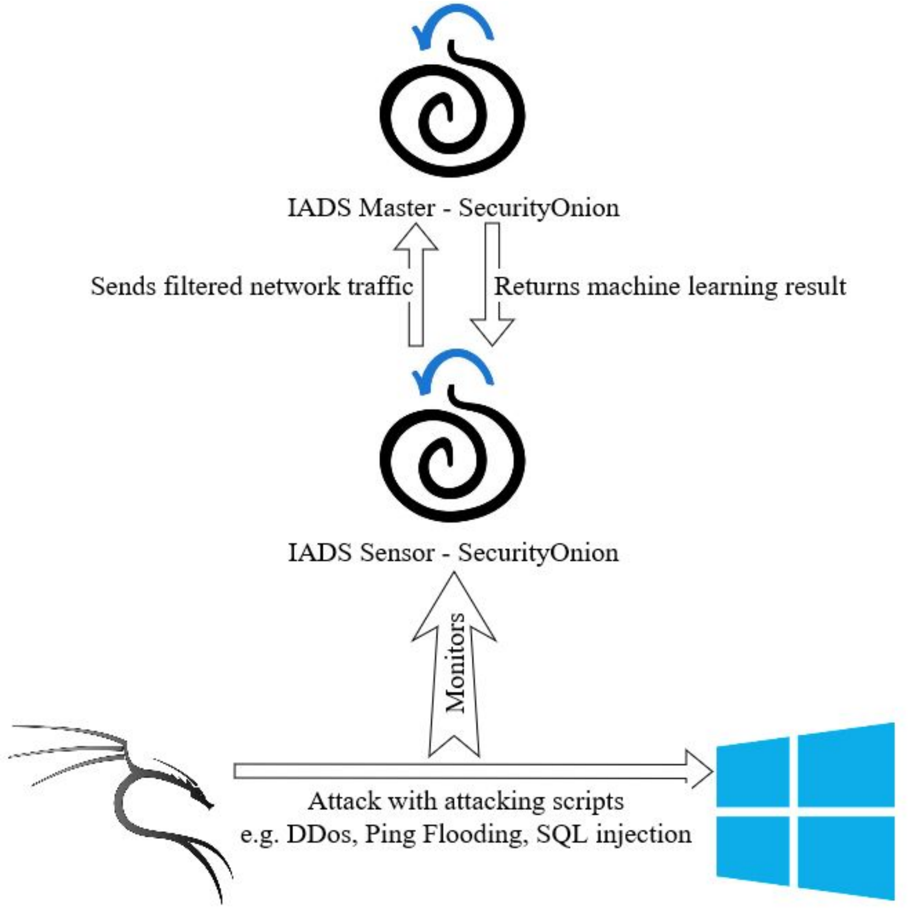
Detailed Diagram

- Our design accepts many alerts as input, originating from a variety of sensors
- These alerts will be processed, ensuring that they enter the anomaly detection system
- Then, our algorithm will perform alert correlation through the Cloud
- Our machine learning model will be trained by a dataset to draw relationships between alerts
- The conclusions will then be visualized for user analysis

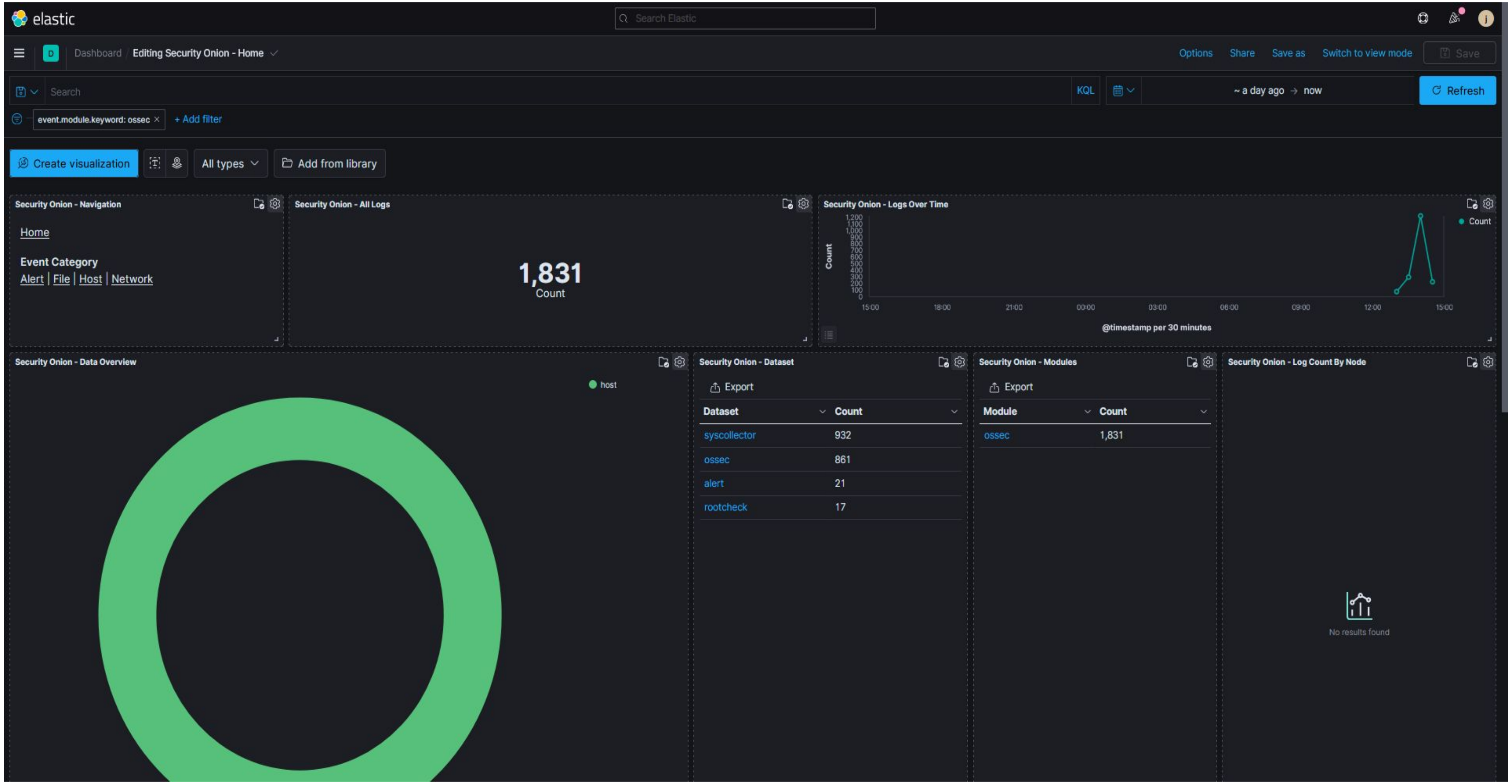


Prototype Implementations

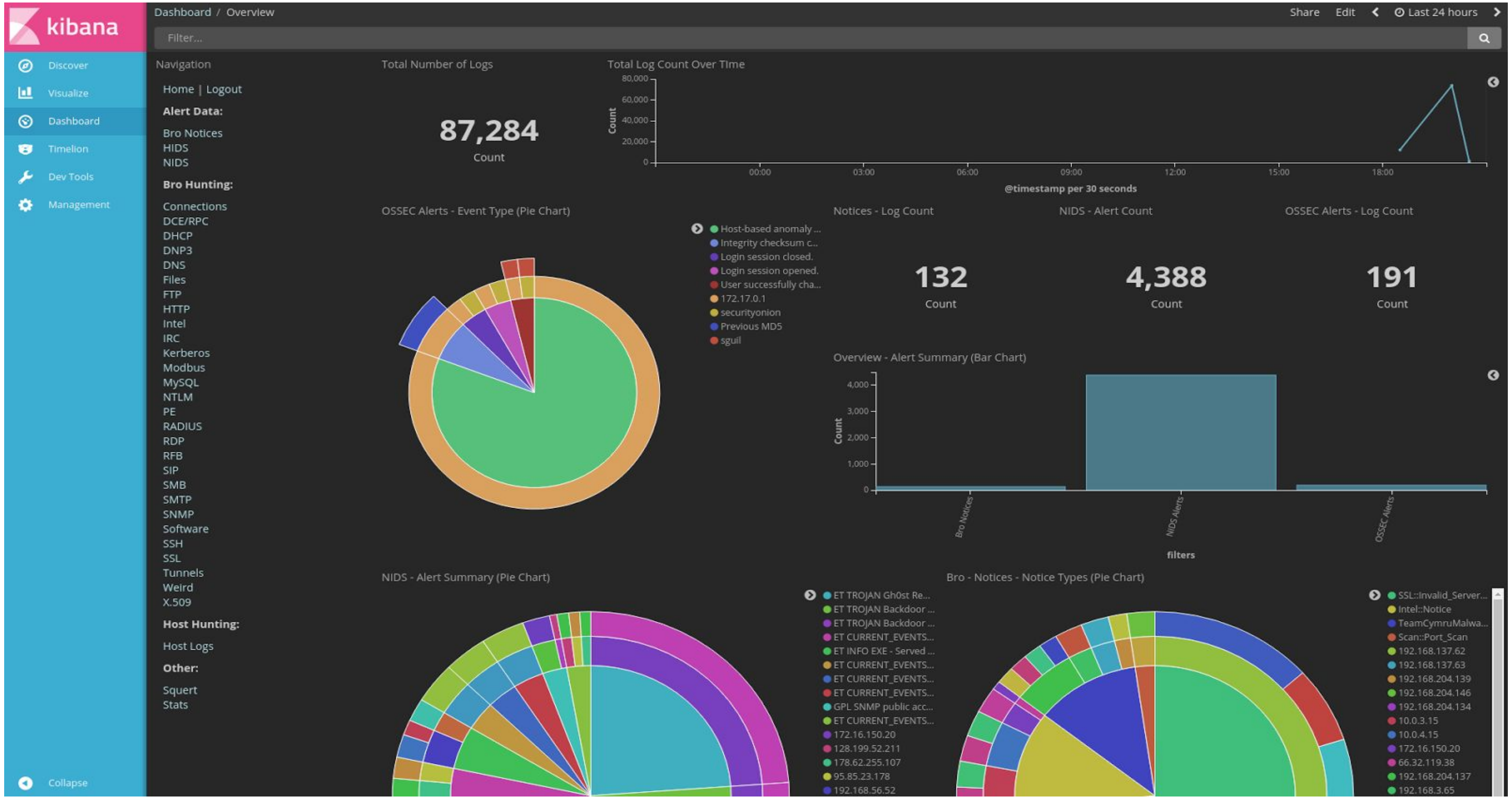




IADS Master Dashboard



Visualizing Alerts in Kibana



https://blog.securityonion.net/2017/06/towards-elastic-on-security-onion.html

NSL - KDD Dataset

No.	Feature Name	No.	Feature Name
1	Duration	22	is_guest_login
2	protocol type	23	**count**
3	service	24	srv_count
4	**flag**	25	**serror_rate**
5	src_bytes	26	**srv_serror_rate**
6	**dst_bytes**	27	**rerror_rate**
7	land	28	srv_rerror_rate
8	wrong_fragment	29	**same_srv_rate**
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	**dst_host_srv_count**
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	**dst_host_serror_rate**
18	num_shells	39	**dst_host_srv_serror_rate**
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	**Classification of the data**

Dataset	Number of Records:					
	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+20%	25192	13449 (53%)	9234 (37%)	2289 (9.16%)	11 (0.04%)	209 (0.8%)

Category	Training Dataset KDDTrain+_20Percent	
	Attack	**Count**
Normal	normal	13,449
Subtotal		13,449
Probe	ipsweep	710
	satan	691
	portsweep	587
	nmap	301
Subtotal		2289
DoS	neptune	8282
	smurf	529
	back	196
	teardrop	188
	pod	38
	land	1
Subtotal		9234
U2R	buffer_overflow	6
	rootkit	4
	loadmodule	1
Subtotal		11
R2L	guess_passwd	10
	warezmaster	7
	imap	5
	multihop	2
	phf	2
	ftp_write	1
	spy	1
	warezclient	181
Subtotal		209
Total		25,192

Model evaluation

Classification evaluation docs

Job status: stopped

docs evaluated: 25192

Normalized confusion matrix for entire dataset

Predicted class

Columns

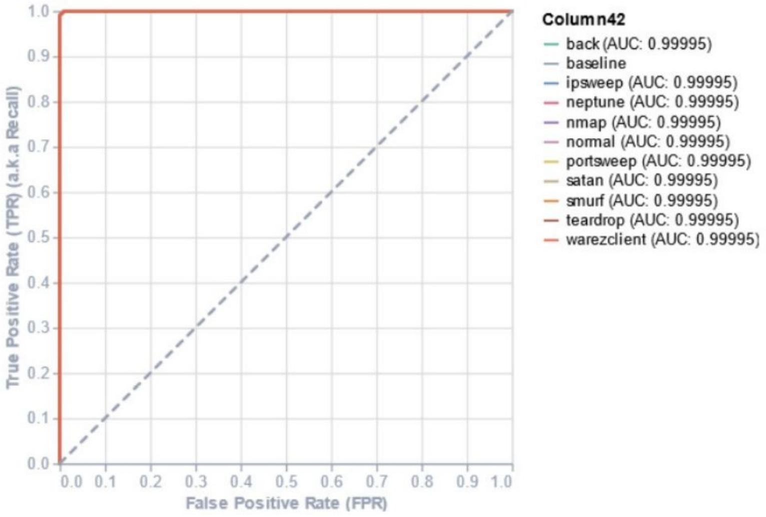
Actual class	back	ipsweep	neptune	nmap	normal	portsweep	4 more
back	100%	0%	0%	0%	0%	0%	...
ipsweep	0%	100%	0%	0%	0%	0%	...
neptune	0%	0%	100%	0%	0%	0%	...
nmap	0%	0%	0%	100%	0%	0%	...
normal	0%	0%	0%	0%	100%	0%	...
portsweep	0%	0%	0%	0%	0%	100%	...

Evaluation quality metrics

0.999 Overall accuracy

0.948 Mean recall

Receiver operating characteristic (ROC) curve



Testing and Evaluation

- Testing is performed on the actual platform (Client VM)
- Functional testing
  - JUnit, Mockito - UI testing
  - Visualization testing
- Security evaluation
  - Black box testing - Low false positive, false negative rate
- Timing evaluation
  - Real-time operation

Design Complexity

- Our design required understanding of advanced technical concepts such as anomaly detection, network intrusion, and machine learning algorithms. Since none of us had any background in cybersecurity, this caused us to spend an unexpected amount of time conducting background research.
  - The required testbed environment was unreliable.
- Design iterations needed:
  - Local implementation vs testbed implementation
  - Datasets
  - Algorithms
  - Single Sensor

Project Plan – Milestones

- Filter and track alerts through spatial and temporal data
- Accept continuous alerts from the sensor VMs
- Build a UI to display tracked anomaly data intuitively
- Process every alerts in 10 ms delay for real time operation
- Eliminate false positives with 90% accuracy

Next Steps Spring 2022

GANTT CHART

PROJECT TITLE	SDMAY22-38	COMPANY NAME	SDMAY22-38
PROJECT MANAGER	Manimaran Govindarasu	DATE	12/05/21

PHASE ONE	PHASE TWO	PHASE THREE	PHASE FOUR	PHASE FIVE	PHASE SIX	PHASE SEVEN
WEEK 1-2	WEEK 3-4	WEEK 5-6	WEEK 7-8	WEEK 9-10	WEEK 11-12	WEEK 13-14
M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F

TASK TITLE	TASK OWNER	START DATE	DUE DATE
Sprint 1			
Set up IADS Sensor VM to provide filtered information to the IADS Master using Snort	All	1/14/22	1/28/22
Sprint 2			
Test the connection between the Sensor and the Master using SecurityOnion. Data augmentation using existing open-source dataset as well as actual data.	All	1/31/22	2/11/22
Sprint 3			
Use selected machine learning algorithm to detect anomalies from the data.	All	2/14/22	2/25/22
Sprint 4			
Compare different machine learning algorithms and refactor the algorithm if needed.	All	2/28/22	3/11/22
Sprint 5			
Put IADS Master in the cloud and test out the alert correlation algorithms.	All	3/21/22	4/1/22
Sprint 6			
Build the User Interface in the Cloud including data visualization.	All	4/4/22	4/15/22
Sprint 7			
Review and test the system.	All	4/18/22	4/29/22

Conclusion

We aim to use a machine learning algorithm within our VM to process alerts, ensuring that they contain all the necessary information such as time, location, severity, and type to be analyzed by a user. We will go into the future of this project with the ultimate goal of designing a user experience that emphasizes accuracy, speed, and utility.

Thank You For Your Time!

Any Questions?

References

Slide 13 Image

https://blog.securityonion.net/2017/06/towards-elastic-on-security-onion.html