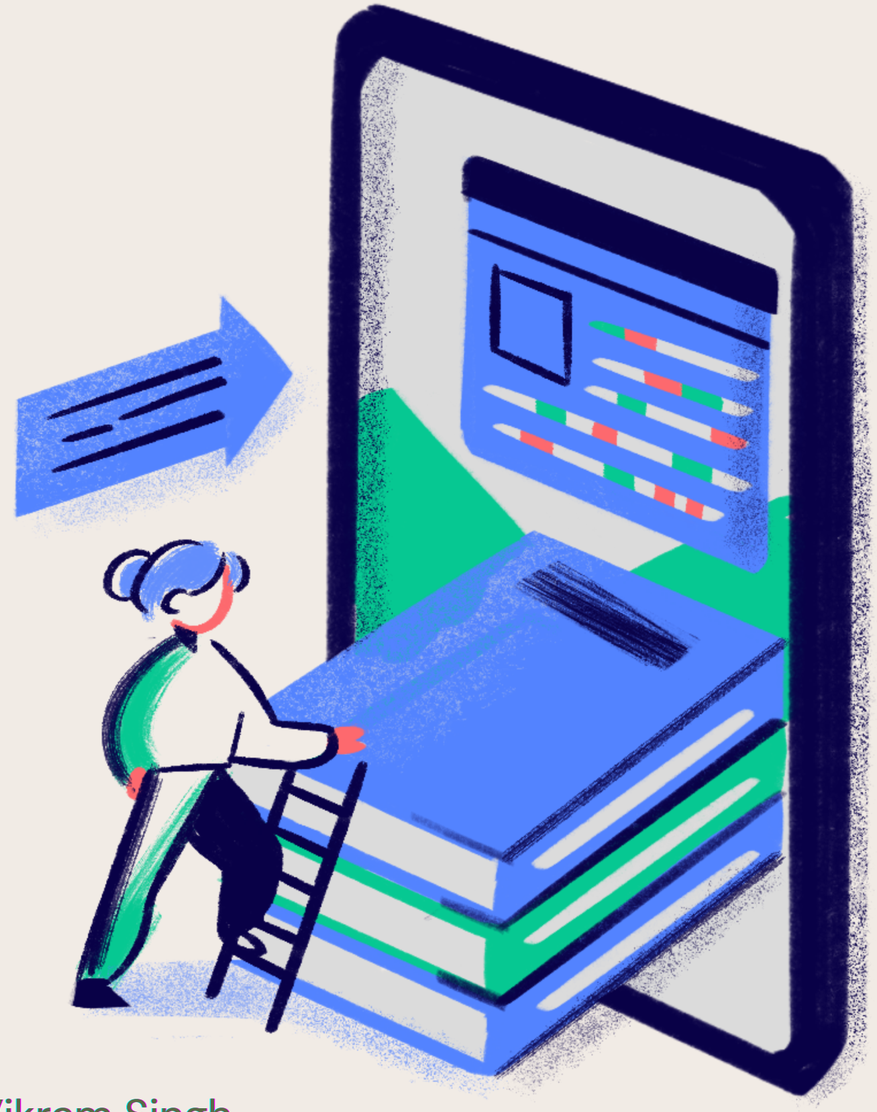
Group-11

Advanced Botnet Detection

Using PSI graph



Aditya Pote
2020IMT-069

Shikhar Gupta
2020IMT-090

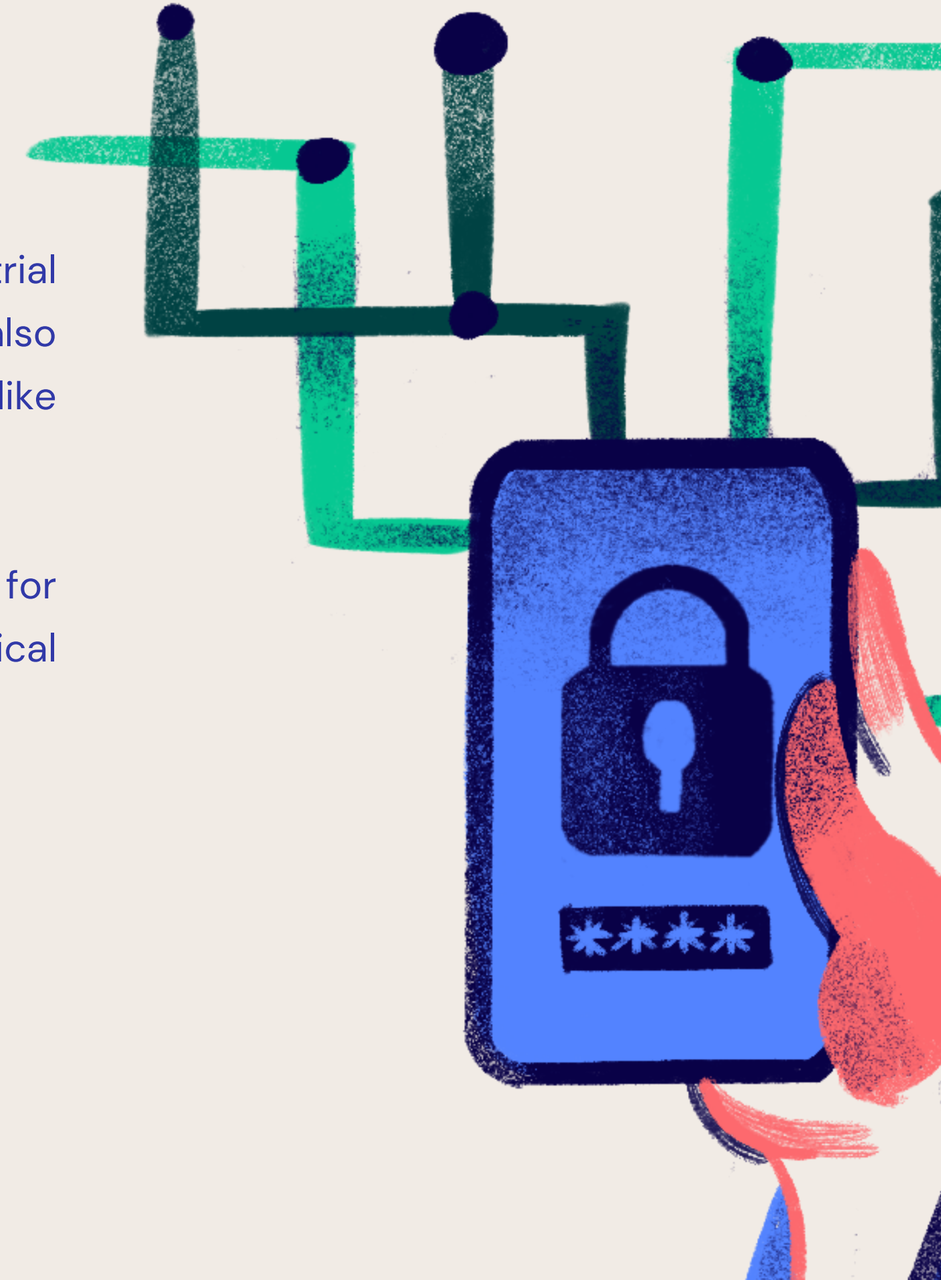
Suyash Vikram Singh
2020IMT-104

introduction

1. The rapid advancement in technologies such as AI, Big Data, and the Industrial Internet of Things (IIoT) has significantly enhanced various sectors but also introduced substantial cybersecurity risks, particularly from IoT malware like botnets.

2. With IoT devices projected to reach nearly 75 billion by 2025, the urgency for robust IoT malware detection mechanisms has intensified, highlighting the critical challenge of safeguarding device security against escalating cyberattacks.

3. We need a novel approach involving multiple techniques to mitigate this issue.



Challenges in botnet detection

Evolving Threats

Botnets keep changing tactics, making detection methods outdated quickly

Limited Resources

IoT devices often have limited processing power, memory, and battery life.

Encrypted Traffic

Attackers may encrypt botnet traffic to make it more difficult to detect.

Literature Survey

Paper	Results	Limitations
A novel graph-based approach for IoT botnet detection	• Achieved 98.7% accuracy in detecting IoT botnets. • Outperformed existing methods in IoT botnet detection.	• Challenges in addressing multi-architecture issues on IoT devices. • High computational resource consumption for analyzing IoT botnets.
A Review on Attack Graph Analysis for IoT Vulnerability Assessment	• The survey highlights core modeling techniques for IoT vulnerability assessment. • Identified methodologies and technologies used in generating and analyzing attack graphs.	• Lack of standardized IoT networks for realistic vulnerability assessment. • Challenges in autonomous attack graph generation for larger systems.
An Automated and Comprehensive Framework for IoT Botnet Detection and Analysis (IoT-BDA)	• Captured, analyzed, and reported 4077 unique IoT botnet samples. • Identified anti-analysis, persistence, and anti-forensics techniques in IoT botnets.	• Lack of integration between capturing and analyzing IoT botnet samples. • Effectiveness of sandboxes limited by anti-analysis techniques.

Methodology

01.

Dynamic Analysis

Execute the .exe or .elf files in an controlled environment and monitor the system calls, network calls, etc. A graph is created during this dynamic analysis showing the

02.

Static Analysis

Use Delinker to convert the .exe file back to the assembly code. Another graph is created in this static phase.

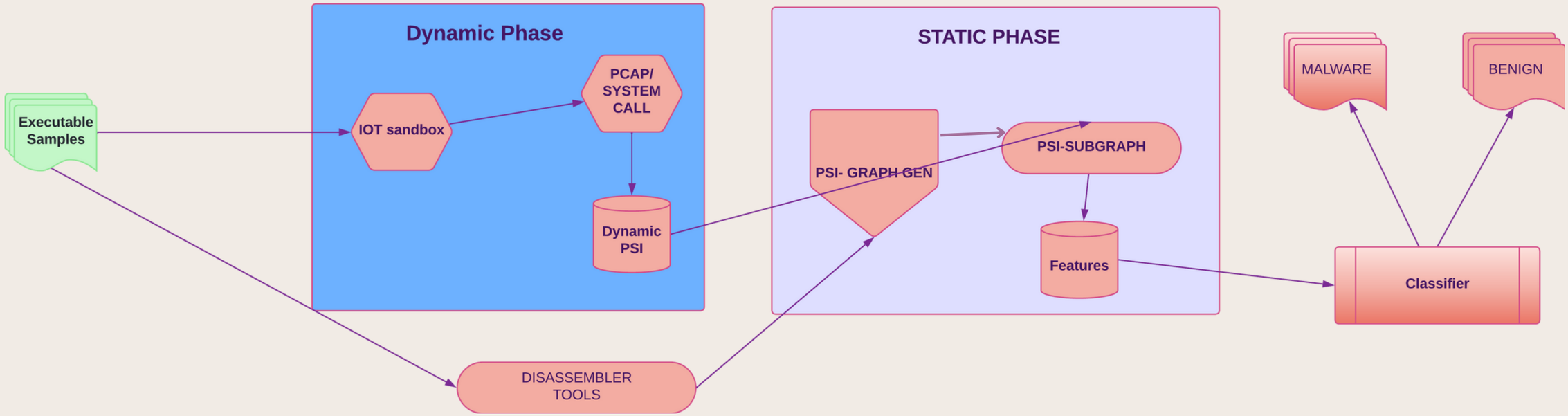
The two graphs are then combined to get a final graph.

03.

Feature extraction and classification

The final graph obtained is converted to a feature vector using various graph algorithms present. Finally, ML classification techniques are employed to mark the file as ‘safe’ or ‘unsafe’

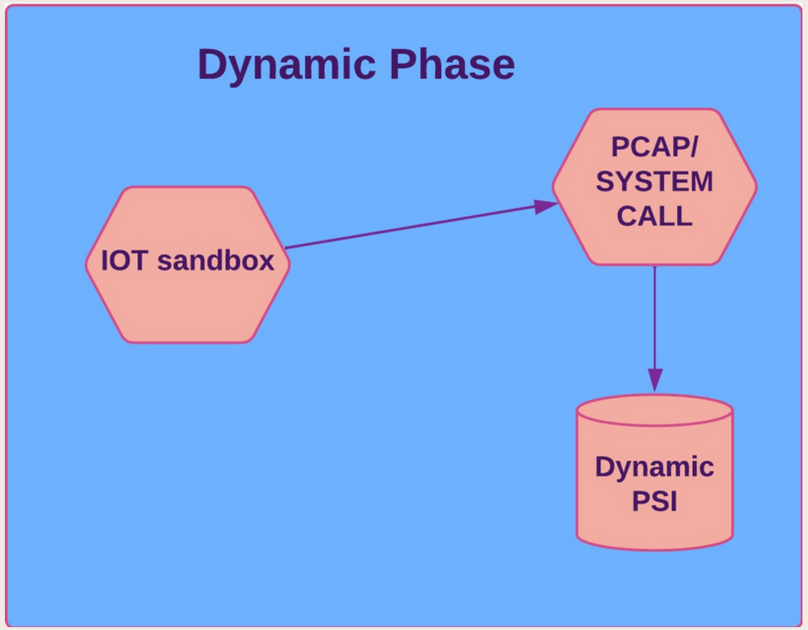
Architecture



Architecture

Dynamic Phase

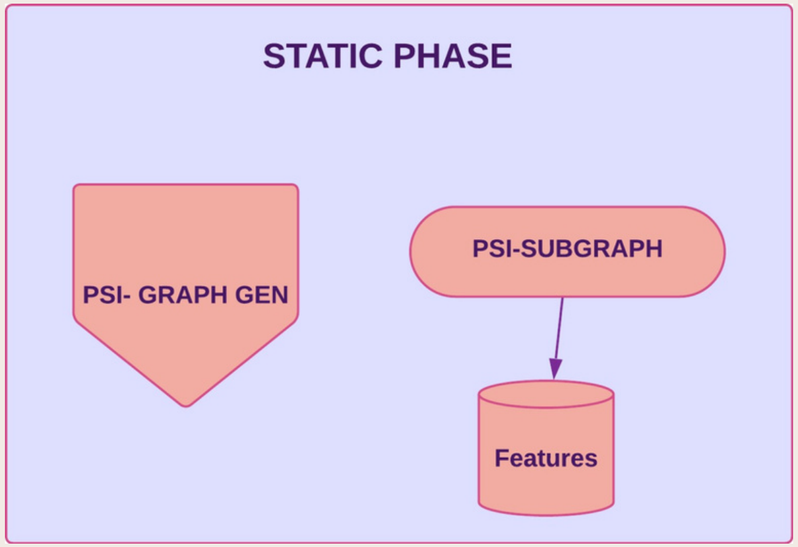
- **IoT Sandbox Execution**: IoT samples are executed in a controlled environment called an IoT Sandbox. This is where the dynamic analysis takes place.
- **Data Generation**: During execution, the samples generate network traffic (PCAP) and system calls, which are indicative of the sample's behaviour.
- **Dynamic PSI Creation**: The generated data is used to create a Printable String Information (PSI), which is a dynamic representation of the sample's behaviour.



Architecture

Static Phase

- **PSI-Graph Generation**: A PSI-graph is generated from the Dynamic PSI. This graph represents the static structure of the sample's code.
- **Combination of the two graphs**: The two subgraphs are then combined to get a final graph based on standard system network calls so as to remove unnecessary calls and noise.
- **Feature Extraction**: Specific features are extracted from these subgraphs, which are crucial for the next step of classification.
- **Classification**: The extracted features are fed into a classifier. The classifier uses machine learning algorithms to determine whether the sample is malware or benign based on the features.



Architecture

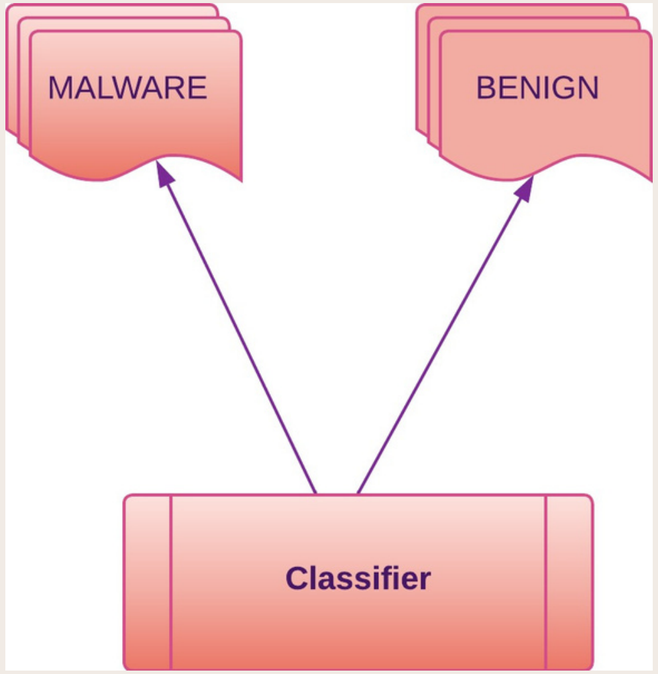
Feature Vector Conversion:

- The graph-to-feature vector algorithm is designed to learn latent representations of rooted subgraphs from large graphs.
- It encodes semantic substructure dependencies in a continuous vector space, which statistical models can use for various tasks, including classification.
- This process involves taking the PSI-rooted subgraphs and transforming them into a numerical form that represents the structural and behavioural features of the traffic samples.

Architecture

Classification

- Once we have the feature vectors, you can apply classification techniques.
- The classification can be performed using machine learning algorithms such as Random Forest, Decision Tree, Bagging, k-nearest Neighbor, and Support Vector Machine.
- These classifiers will use the feature vectors to train a model that can distinguish between benign and malicious IoT samples with high accuracy.



PRESENTED BY GROUP-11

THANK YOU VERY MUCH!