Breach Prediction
abhishek Singh

FLow of Presentation

- **Deterministic Analytics**
- **AI Leaning Models**
  - Data Set for the Models.
  - Model which is best suited.
  - Proposed Architecture which be used to test POC, ship it.
  - Risk Factors

GUI for Breach Prediction

- **Severity:** High, Medium , Low
- **Reason**: Password Compromise,
- **Remedial Action**:
  - Apply CASB Policies
  - Change Password
  - Drop Network Connection
- **Asset:** Server, Person (Remediation Purposes)
- **Possible Impact** : Exfiltration, Account Access Removal, Data Encrypted for Impact, Data manipulation, Defacement, Disk Wipes, EndPoint Denial of Service, Firmware Corruption, Inhibit system Recovery, Network Denial of Service, Resource Hijacking, Service Stop, System Shutdown/Reboot, Exfiltration.

GUI for Breach Prediction

- **Severity:** High, Medium , Low
- **Reason**: Password Compromise,
- **Remedial Action**:
  - Apply CASB Policies
  - Change Password
  - Drop Network Connection
- **Asset:** Server, Person (Remediation Purposes)
- **Possible Impact** : Exfiltration, Account Access Removal, Data Encrypted for Impact, Data manipulation, Defacement, Disk Wipes, EndPoint Denial of Service, Firmware Corruption, Inhibit system Recovery, Network Denial of Service, Resource Hijacking, Service Stop, System Shutdown/Reboot, Exfiltration.

High Risk Deterministic Analytics for Breach Prediction

- **Password Leakage / Login Attempts from Compromised Passwords:**

  - Retrospective Phishing URLs Verdicts (VM / ML based solution, Third Party Feeds ….) correlated with the ZTNA (/Web GateWay) logs

    * Retrospective Phishing Links (North South Traffic), POST request has been generated = Passwords Compromised

    * Retrospective Phishing Links (East-West / Internal Traffic) + Send to many Recipients + POST request has been generated = Internal Account which has send phishing links has been compromised.

  - 0365 Audit Log events: ***UserAgent*** (User's Browser Information) and ***Client IP Address***

    Client IP Address is the IP Address from where a person is logging

    Anomaly Detection using GMM: Features IP (GeoLite Database ASN, Country, City), User Agent

    Rule Based: IP & User Agent both changes = Breach

Breach Predictor Analytics: Medium Risk

Exploit Public Facing Application

Vulnerable Applications

Web Gateway logs to determine vulnerable application, which has been used by threat actors. Below are some of these.   Here we have to use banner information to get version of Software installed.

- Zoho Manage File System (CVE-2022-35405)
- Citrix ADC & Citrix GateWay (CVE-2022-27518)
- Microsoft Exchange and Support Diagnostic tool (CVE-2022-41040 & CVE-2022-41082)
- VMware vCenter Server (CVE-2021-22005), WS02(CVE-2022-29464),
- Apache Log4js, F5 Big IP Device (CVE-2022-1388)

Detection of WebShells

- Outbound traffic for detection of webshells

Deep Learning Transformer Based Neural Networks Model Sources to Extract

Sources to Extract

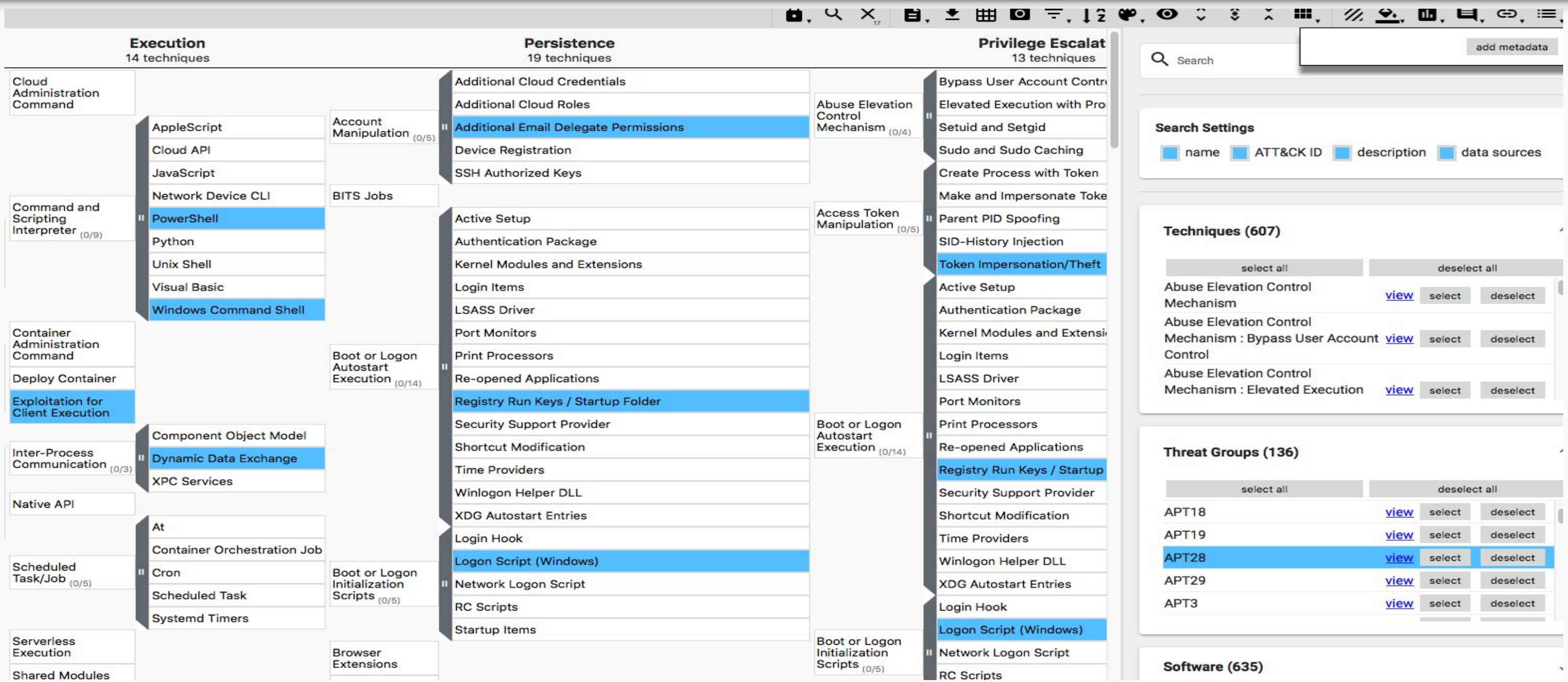
Training Data: (Structured Data align it as per the Mitre Techniques and Tactics it captures sequence of steps)

- Techniques and Tactics of Threat Actor known breaches from Mitre Navigator
- DataBase Alert Trigger from Deception Logs:
  * Multi Stage Malware
  * Threat Actor & Mapped them to Mitre Tactics and Techniques
- DataBase: Malicious C&C network communication from compromised endpoints/server and gather indicators before that.
- DataBase: For each of the data set gather additional feature set : Industry Segment, Time of day, Day of Week , Geolocation, User Behavior, version of software…. (Reduce False Positives / Train the Transformer based Model)
- Type of Source: Deception

Data Source:

- Network (Email, Web,SMB,RDP,)
- Deception
- EDR Logs
- Active Directory Logs
- Cloud Logs, Container, SaaS, IaaS

Mitre Navigator



Neural Network Model Transformer Based Model

- Structured Data align it as per the Mitre Techniques and Tactics it captures sequence of steps
  Actual Attack can be dynamic. Compute variations model it as per Mitre technique and Tactics

- Represent each observation or event as Mitre Technique and Tactics
  (Reconnaissance, Resource Development, Initial Access, Execution, Persistence,
  Privilege Escalation, Defense Evasion, credential Access, Discovery, Lateral Movement)

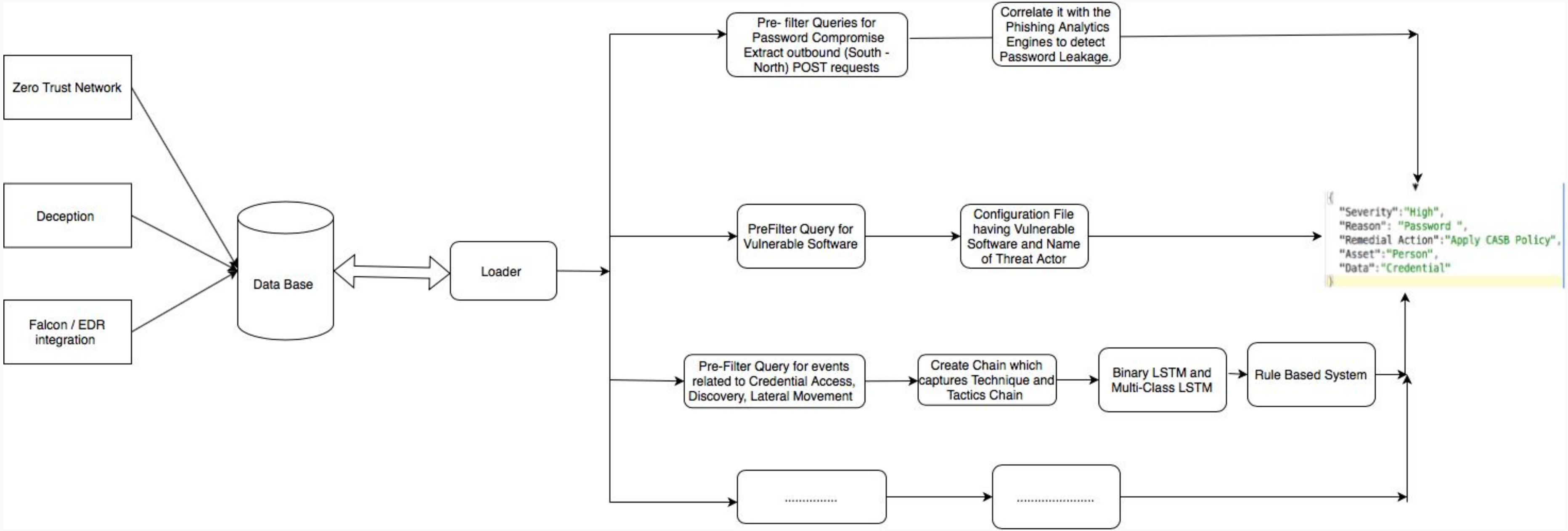
- Convert data as a sequence of tokens, add embeddings which can be fed into the model.

- Train the Transformer based model for sequences of techniques and tactics to predict breach.
  Sigmoid Activation Function : Breach or Non Breach. (Target: Collection, Command & Control, Exfiltration, Impact).
  Softmax Activation Function: Type of Breach, APT1, APT2, Malware ..

- Greater than threshold we mark it as malicious for less than threshold combine it with other indicators.

- Since the sequence capturing semantics of Mitre Technique and Tactics is small, LSTM may work better than
  Transformer based model.

Proposed Architecture to Implement / Test POC



Risk Factor

- Neural Network is heavily dependent on the Data Benign and Malicious which can take reasonable time.

  - Mitigation for V1 the output of model can be combined with additional indications to reduce false positives. (Industry Segment, Time of Day, IP, etc..),

  - Feedback Loop from Customer can also be build which will retrain the data set.

  - Not always Structured can be dynamic. For such cases use Rule based conditions to correlate the conditions and test the feasibility of the malicious and identify structure, get labelled sample set for input for Neural Network Model.

- Transformer model works well if tested on actual production traffic: The proposed designed architecture will be integrated with the DataBase and will be tested on actual production traffic.

- The model has to be fast before the exfiltration or Impact stage is reached. It will be trained to predict **Collection, Command & Control, Exfiltration, Impact.**

Appendix Sample code.

```python
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Example MITRE ATT&CK steps data (replace with your actual data)
mitre_steps_data = [
    [1, 2, 3],
    [1, 4],
    # ... more data ...
]

# Example breach labels (replace with your actual labels)
breach_labels = [1, 0, 1, 0, ...]  # 1: Breach, 0: No Breach
# Convert MITRE steps data to sequences of numerical representations
data_sequences = mitre_steps_data
# Create a vocabulary for the embedding
vocab = set()
for sequence in data_sequences:
    vocab.update(sequence)
vocab_size = len(vocab)

# Pad sequences to a fixed length (you can adjust this)
max_sequence_length = max(len(seq) for seq in data_sequences)
padded_sequences = [seq + [0] * (max_sequence_length - len(seq)) for seq in data_sequences]

# Convert data to PyTorch tensors
X = torch.tensor(padded_sequences, dtype=torch.long)
y = torch.tensor(breach_labels, dtype=torch.float32)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
# Define the Transformer-based breach prediction model
class TransformerBreachModel(nn.Module):
    def __init__(self, input_vocab_size, embedding_dim, nhead, num_encoder_layers):
        super(TransformerBreachModel, self).__init__()
        self.embedding = nn.Embedding(input_vocab_size, embedding_dim)
        self.transformer = nn.Transformer(
            d_model=embedding_dim,
            nhead=nhead,
            num_encoder_layers=num_encoder_layers
        )
        self.fc = nn.Linear(embedding_dim, 1)

    def forward(self, x):
        embedded = self.embedding(x)
        output = self.transformer(embedded, embedded)
        output = self.fc(output.mean(dim=1))
        return output

# Instantiate the Transformer-based breach prediction model
embedding_dim = 128
nhead = 4
num_encoder_layers = 2

model = TransformerBreachModel(vocab_size, embedding_dim, nhead, num_encoder_layers)

# Loss function and optimizer
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 10
batch_size = 16

for epoch in range(num_epochs):
    model.train()
    for i in range(0, len(X_train), batch_size):
        batch_X = X_train[i:i+batch_size]
        batch_y = y_train[i:i+batch_size]

        optimizer.zero_grad()
        outputs = model(batch_X)

        loss = criterion(outputs.view(-1), batch_y)
        loss.backward()
        optimizer.step()

    # Evaluate on the test set
    model.eval()
    with torch.no_grad():
        test_outputs = model(X_test)
        test_outputs_rounded = torch.round(torch.sigmoid(test_outputs))

        accuracy = accuracy_score(y_test, test_outputs_rounded)
        print(f'Epoch [{epoch+1}/{num_epochs}], Test Accuracy: {accuracy:.4f}')
```

Deception Logs Malware

Threat Class	Malware in the Threat Class	Detection Stage as per Mitre Threat Matrix	Condition leading to the detection of breach	Breadcrumbs /Lures on the endpoint	Deception on the internal network
Destructive Malware	Shamoon, Olympic Destroyer, Petya	Lateral Movement	Brute force attempt, detection of RCE, usage of honey credentials to log on to SMB deceptions, SMB exploit class packets	Entry of deceptions such as SMB, DB, in the ARP table, Established connection to deceptions in network. Honey credentials in lsass.exe.	Projected SMB deceptions in the subnet
Ransomware	99% of the Families	Execution Phase	Encryption of Honey files will trigger proprietary algorithm.	Honey Files, Mapped Drives	Honey Unmapped Drives.
Cryptominer	WannaMine, Zealot campaign, ReddisWannaMine	Lateral Movement Phase	Network scanning, detection of RCE, usage of compromised credentials., SMB exploit class packets	Established connection from every network adaptor of Web Server, Endpoint to SMB, DB, FTP deceptions. Honey credentials in lsass.exe	Deceptions such as SMB, DB, FTP in the subnet. Deceptions having Class B IPv4 address.
Information Stealer	Emotet, Qakbot	Lateral Movement Phase	Brute force attempts, usage of compromised credentials.	Honey username and deception services in AD. Honey credential on the endpoint. Honey email address in Outlook	Projected SMB deceptions in the subnet
Breaches involving Web Server	Remote code Apache Struts e.g. CVE-2017-5639, CVE-2017-9822, WebShells	Lateral Movement Phase	Detection of Scan originating from Web Server, detection of RCE, usage of honey credentials to log on to deceptions, brute force attempts, SMB exploit class packets	Established connection from Webserver to Deceptions. Honey Password in lsass.exe	DB, SMB, FTP deception reachable from Webserver, Deceptions having class B IPv4 addresses.
Password Stealer	Pony password stealer, Ovidiy password stealer	Execution Phase	Usage of compromised password to log on to honey services such as SQL, FTP, SMB	Honey credentials in the browser, Registry Entries and at specific locations	Deceptions such as SMB, DB, FTP in the network.

Table 1.0 Showing the detection of Critical Threats using Distributed Deception.

Deception Logs Threat Actor

Phase at which the threat will get detected as per the Mitre Threat Matrix	BreadCrumbs and Lures which are required at the end point	Condition leading to the detection of Breach	Deception at the Network	Threat Actor / Breaches which could have been diverted to the engagement platform.
Lateral Movement Phase	Honey Mapped Drives.	Accessing Files Honey mapped Drives in a short span of time.	Services such as databases, SMB in the network.	OrangeWorm [1] (Hospital Breaches), Monsoon[9], Leviathan[10]
Execution Phase	Honey Credential of services in the Browser, Keychain, files. Honey Credentials in LSASS.	Usage of the deception credentials in the network.	Services such as DB, FTP, SMB in the network	APT 37 (ZUMKONG Malware)[2], Bronze Butler [3], Cleaver[7], Muddy water[8], APT 28[4], Cozy Duke [5], APT 34, APT 32 [6], Stealth Falcon[12]
Lateral Movement Phase	Entries of the deception in the networks in the ARP cache.	Sending Remote code exploits, scans, compromised passwords, brute force attempts to the services in the network	Services such as databases, FTP, SMB in the network.	Stealth Falcon[12], Orange Worm[1], Strider[13]

Table 1. 0 Showing the diversion of breaches by using breadcrumbs at the endpoint.