USER BEHAVIOR ANALYSIS OVER A DYNAMIC IMPLEMENTED ZERO TRUST NETWORK ARCHITECTURE

Greta Germizi[1], Wassim Ahmad[2]

[1] Department of Software Engineering, Faculty of Engineering, Canadian Institute of Technology, Albania, greta.germizi@cit.edu.al, ORCID: 0009-0009-2095-8922

[2] Department of Software Engineering, Faculty of Engineering, Canadian Institute of Technology, Albania, wassim.ahmad@cit.edu.al, ORCID: 0009-0009-8976-1976

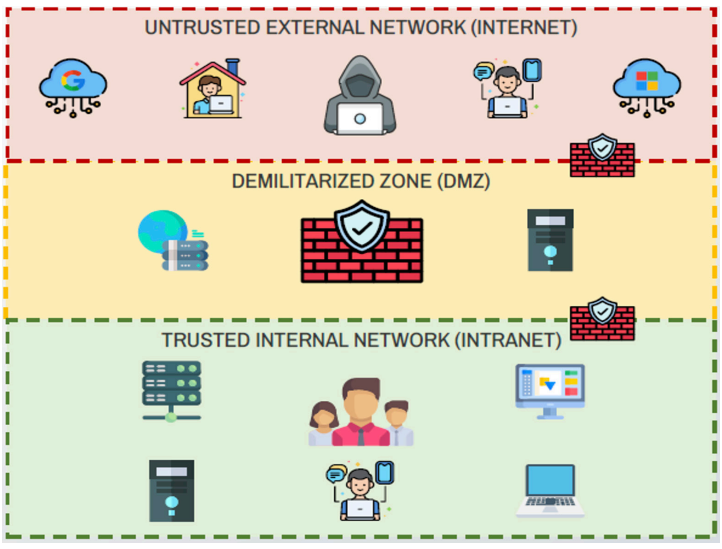
Abstract

The dynamics of the ever-evolving threat landscape in the world of cyber keep advancing at such a speed that it is crucial for the organizations to adopt innovative security frameworks. However, the question that arises is how accessible are these frameworks for the organizations? This paper aims to highlight the importance of implementing a Zero Trust Network Architecture as a means to address all these security challenges, in a way that can be accessible for every organization regardless of size and without any significant financial constraint. While previous research examines the ZTNA framework and its applications, alongside there are other studies about User Behavior Analysis and its implications in cybersecurity focusing on theoretical aspects of each. Nevertheless, the solution this paper presents, incorporates both of them into a practical implementation that aims to improve organizational security posture. Yet, how will be reduced such an attack surface, with the risk posed not only on the outside perimeter but as well as by their own employees? By leveraging machine learning algorithms, we aspire to demonstrate that a robust tool fed by ZTNA logs could be created providing real-time threat detection and automated response mechanisms thereby reducing the possibility of lateral movement inside the infrastructure and colossal damage or theft of data and accesses. It comprises the usage of solely open-source tools avoiding vendor lock-in and allowing more flexibility and accessibility on integration of various technologies. Key findings recommend constantly updating baseline profile for accuracy in anomaly detection. The research illustrates that comprehensive security measures are achievable for all organizations, ensuring enhanced protection in today's dynamic network environments.

Keywords: Zero Trust Network Architecture (ZTNA), Dynamic Framework, Cybersecurity Challenges, Machine Learning Algorithms, Threat Intelligence, Continuous Authentication

INTRODUCTION

In recent years, the cybersecurity landscape has witnessed a rise in sophisticated attacks targeting organizations worldwide. It is particularly alarming to note the rise in insider threats, in which workers or other authorized personnel inadvertently or purposely jeopardize data security. Based on the statistics from the Data Exposure Report (DER), one in three data breaches are caused by insider activity, counting an average of $16 million annual financial loss per incident. (Code 42's, 2023) Considering the fact that insiders can jump off the security measures designed for the outsiders not to break in the system, it becomes challenging to identify these actors and stop their actions. According to the Cost of Insider Threats report, it will take an estimated 85 days to suppress the threat. This makes insider threats particularly difficult to counter. The longer duration not only increases organizational costs but also complicates the containment process because more resources have to be dedicated to handling these situations. Marking digital transformation as the main catalyst, there are a number of additional variables that make today's infrastructure and network architecture vulnerable to these threats. IT infrastructure used to be composed of a firewall-protected trustworthy internal network (intranet) and an untrusted external network (internet) as depicted in Figure 1. But as technology developed, this boundary became increasingly hazy with the emergence of DMZs for services that are visible to the public, such as websites.



1. Traditional Network Perimeter (Alton Teaches- Udemy Course ZTNA Principles)

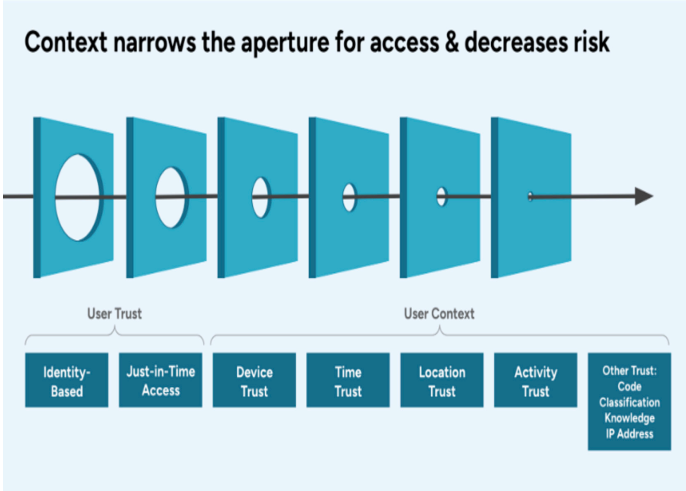
*Corresponding author

Moreover, as employees started accessing their company internal network from several different locations and devices, lines were slowly blurring where concepts like remote work and BYOD became the new trend. Zippia, the career expert, conducted a survey and stated that 83% of businesses had brought in bring your own device (BYOD) rules by 2023, permitting employees to use their own devices on company networks. The old concept of the trust boundary was eroded daily, as the evidence brings up the observations that 8 out of 10 people now work remotely or have hybrid schedules, based on a 2023 Gallup survey.

Observing the existing infrastructure of most of the enterprises and how they manage remote connection for client applications or employee access, it results that 72% of them use VPN connections. (Nicoletti, 2022) VPN's pose greater risk to the organization, giving unlimited access to the whole network once a user is successfully authenticated. Only with some user credentials and the VPN tunnel a hacker can bypass the access and get sensitive information of the company and the hosts in the network, especially with the advancement of tools and quantum computers that in a matter of seconds can break in the encryption used.

As a result of the growing adoption of cloud services, most of the resources considered as critical for the organization were scattered around the internet not sticking to the old network defined perimeters. According to a 2021 O'Reilly poll, almost 90% of the enterprises had embraced cloud and used their services for hosting the applications and managing their infrastructures. Consequently, the conventional notion of the trust perimeter inside of an on-premise infrastructure vanished, indicating the necessity to transition to a new paradigm. Considering these facts organizations would start to develop a new perspective on treating their environment as hostile and giving up from an implicit level of trust inside their perimeters. They would tentatively need a granular access control mechanism and a proactive approach to threat detection and response. Coming up to these needs of the organizations nowadays, there is a set of principals building upon a whole framework architecture called Zero Trust Network Architecture. In response to this changing threat scenario, following the Zero Trust Network Architecture (ZTNA) model by organizations has emerged as a critical technique for improving security posture. Furthermore, as per Twilio's COVID-19 digital engagement report, the pandemic accelerated digital transformation by six years, outpacing Moore's Law. This acceleration highlights the necessity of adopting the Zero Trust model in order to keep up with the ever-changing digital landscape.

Still, the question is what makes this model a framework to follow and enhance the security of our organization? Well, ZTNA is not a singular technology, it's more of a philosophy and a mindset. It operates on the premise of a hostile environment, treating both internal and external threats with equal vigilance and disregarding network locality as a trust determinant, emphasizing dynamic, context-based access policies. These policies are based on some foundational component called attribute-based access controls that differently from role-based access controls examine data such as time, location, authentication history, device settings, IP address, and communication type, providing a more sophisticated approach on determining dynamically the access a user must have on several components of the network.



2. Kipling methodology for developing ZTA policies (Michaline Todd, StrongDM)

In order to develop robust access policies regarding these attributes, in alignment with ZTNA tenets for the identities of employees or system users in any kind of role they have, organizations use techniques like the Kipling Method, as in Figure 2. ZTNA architectural approach views trust as a vulnerability in information security, asserting that relying on the reliability or integrity of someone or something is inherently unpredictable due to its behavioral nature. (Edo, 2022) Consequently, ZTNA adheres to
the principle of "Never Trust, Always Verify" while concurrently ensuring individuals receive
precisely tailored access, precisely when and where it is needed—no more, no less.

The goal of this paper is to emphasize the importance of implementing Zero Trust Network Architecture for organizations in this era where the actual conditions of the network infrastructure have changed notably and the necessity for reducing the attack surface is great. However, while the practical benefits of ZTNA are increasingly evident, several challenges remain. Based on the cost-benefit analysis of Zero Trust and Behavior Analytics, the implementation of this framework involves significant costs. (Sharma, 2021). But, how can this be embraced by the organizations wishing to improve their organizational security? By implementing a ZTNA model only by using open-source and vendor-neutral tools we aspire to demonstrate that organizations of all sizes can afford and benefit from this robust security framework. The accessibility of this process makes it feasible for even smaller organizations with limited resources to enhance their security posture effectively. Moreover, the automation of this implementation further simplifies the process, reducing manual effort and ensuring consistent, reliable deployment. Nonetheless, the question that

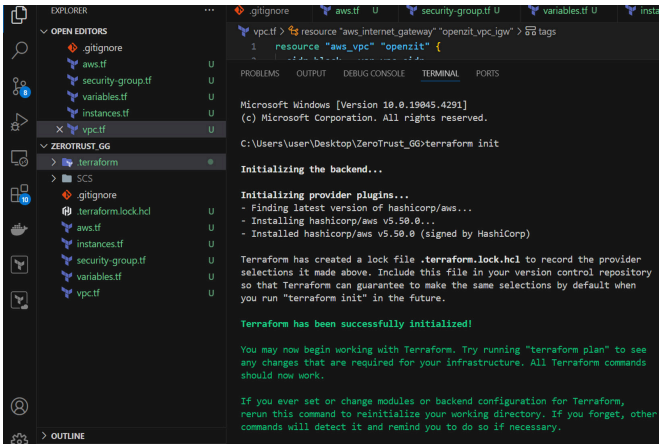
emerges is how scalable is this framework when applied to high-demand network environments? The novelty that we bring is the integration of a machine learning algorithm to analyze user behavior and detect threats in real-time from the logs of the implemented ZTNA architecture adopting to its dynamic nature. This approach leverages advanced analytics to identify and mitigate potential security incidents, enhancing the overall efficacy of the ZTNA framework. Sharma in his research about theoretical implications of the integration of Behavior Analysis to ZTNA, underscores the importance of the baseline profile to achieve desired results in anomaly detection. Still, it falls short in exploring real-time adaptability to evolving user behaviors. This gap is crucial since static baselines may miss nuanced threats in dynamic environments. Addressing this gap, our research will contribute to an adaptive, machine learning-driven behavioral model that continuously updates in response to real-time data. This approach aims to reduce false positives and negatives, enhancing the accuracy and responsiveness of threat detection within ZTNA. This research will illustrate that a comprehensive security approach is achievable for all, regardless of budget or technical expertise, underscoring the practicality and necessity of adopting ZTNA in today's dynamic network environment.

METHODS AND TOOLS

The tools that will be used in this implementation include Terraform for script automation and ease of infrastructure deployment, AWS for infrastructure hosting and VM creation, Open Ziti for applying the Zero Trust principles directly as a tunneler for our application, Docker for the images processed for our web application Odoo Erp, portainer image for having a clear view of the controllers, routers and the state of them, as well as the Ziti Admin Console Interface, Python Libraries and Machine Learning algorithm to detect anomalies in real time for threat detection, Slack for Alert Notifications.

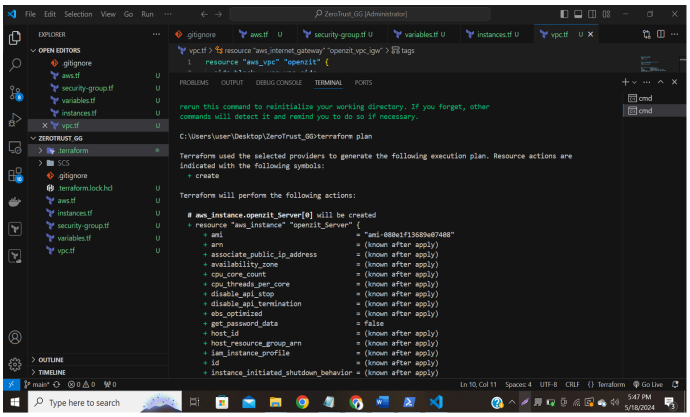
2.1 Creating Infrastructure

The first implementation phase consists of creating the infrastructure needed to deploy this application and embed Zero Trust. The provider we chose to host our Virtual Machines was AWS cloud environment. For creating our VMs that are needed for the web application hosting and openziti servers we continued with an automated solution and create all these VMs all at once with Terraform.



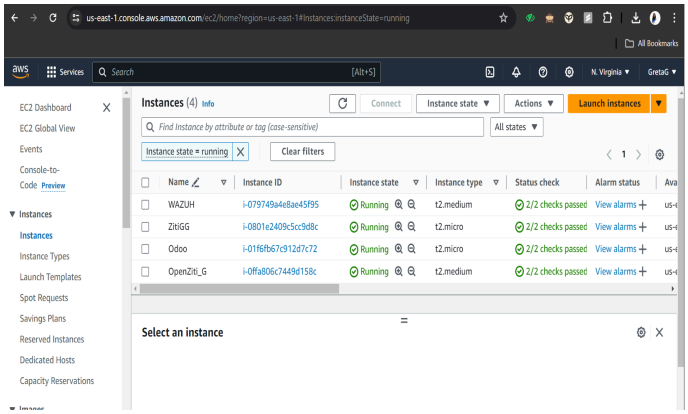
3. Terraform Initialization

After preparing some configuration scripts that detail all the information about the three machines creation including some access and private keys configured in the Key Pair Section in AWS for each service, we initialize Terraform, which prepares the environment for us and downloads all the plugins needed to interact with AWS services. Additionally, it creates a lock file to keep record for the versions of plugins used therefore maintaining consistency for all future runs and promoting reproducibility.



4. Terraform Planning

Proceeding with the planning step after successful initializing terraform, we prepare the scripts for creating 3 VM's all at once, with all configurations predefined. We execute the terraform plan command that starts reading and evaluating all the config files provided and creating an execution plan for them. This way terraform will perform the actions that are planned to achieve the state of infrastructure as we have configured. Additionally, the scripts prepared install docker on every machine automatically without the need to run installation commands one by one in the terminal repetitively for each instance. After application of all these files with the apply command and terraform finishing, we have resources added in our elastic compute dashboard and instances running.



5. AWS Dashboard & Instances Running

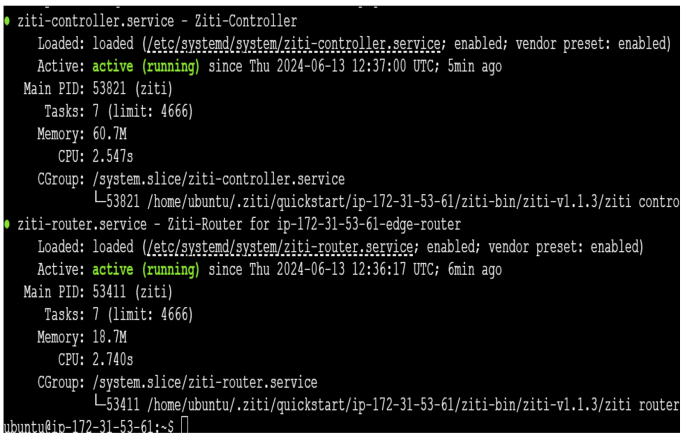
Therefore, we finish with the first step of the implementation phase, where it is demonstrated the efficiency of creating our infrastructure through powerful terraform scripting that allows you to automate the process and write scripts in a declarative way, where you explain how the infrastructure should look like.

2.2 Openziti Configuration

In the second implementation step we focus on configuring OpenZiti environment in order to bring all Zero-Trust principles together in one place. There are several opportunities for setting up OpenZiti and we chose the option that can come in handy for several different cases on which the organization want to set up their network. The case is 'Host OpenZiti Anywhere' that is offered by the Ziti Team with the command of express install and is all opensource. We carry on the process by initializing some variables about the ports where the controller and router will be set up as well as the external public DNS of the instance in AWS where we will set up the OpenZiti 'ec2-100-25-159-180.compute-1.amazonaws.com'. Then executing the command provided in OpenZiti documentation we install all the components needed:

```
source /dev/stdin <<< "$(wget -qO- https://get.openziti.io/ziti-cli-functions.sh)"; expressInstall
```

After this step we verify that the main controller and router are running and active as in the image highlighted below.



6. Controller & Router Status

In order to have an easier way of managing, creating and accessing these routers we can install also the Ziti Admin Console (ZAC) interface that facilitates the process.

Commands used for this:

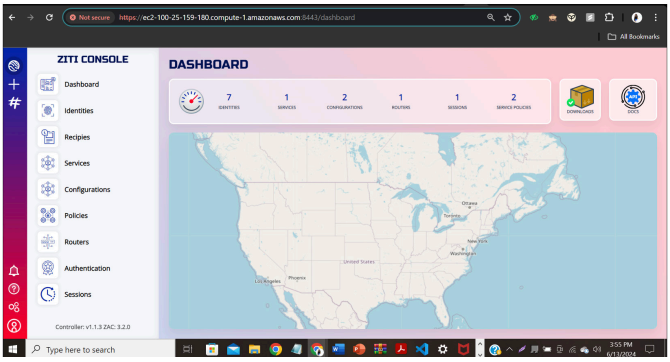
```
sudo apt install nodejs npm -y
sudo npm install -g @angular/cli@16
git clone https://github.com/ openziti/ziti-console.git "$(ZITI_HOME)/ziti-console"
ng build ziti-console-lib node
```

To open the ZAC in a web browser and manage from the interface we need to associate some certificates for this console and we use these commands to do so:

```
ln-s "${ZITI_PKI}/${ZITI_CTRL_EDGE_NAME}-intermediate/certs/${ZITI_CTRL_EDGE_ADVERTISED_ADDRESS}-server.chain.pem" "${ZITI_HOME}/ziti-console/server.chain.pem"

ln-s "${ZITI_PKI}/${ZITI_CTRL_EDGE_ NAME}intermediate/keys/${ZITI_CTRL_EDGE_ADVERTISED_ADDRESS}-server.key" "${ZITI_HOME}/ziti-console/server.key"
```

By doing this we have self-signed certificate assigned to our Ziti Console. This architecture of communication through the Ziti Components but not only, is done also through these certificates using mutual TLS that are used for authentication on both sides of the communication, both parties.



7. Ziti Admin Console Successfully Set Up

Every client should have a client certificate through which it is authenticated and authorized as an identity, that is then checked within Ziti Network if it has access to any application the client is requesting access upon. When successfully checked, all the open ports in the network are closed to ensure that zero access to any open hole in the network is given to the identity, except that of the own application or applications though this identity may have multiple.

2.3 Identities & Services Creation

Coming to the identities in OpenZiti they represent the users that will go through the network filtering of the attribute policies that will be built upon them in order to access the services they need. In order to create identities we used some cli commands but they can be easily created in the interface of ZAC

```
ziti edge create identity user windowsweb -o windowsweb.jwt
ziti edge create identity device ubuvm -o ubuvm.jwt
```

With these commands we created two identities one for the client and one for the server, saving their Json Web Token (JWT) that will be used for authenticating both of the parties. Then we configure our specific network endpoint that will be used for the Odoo service that we will setup on port 8069 with a TCP protocol. The intercept configuration specifies how the traffic to the Odoo service will be handled.

```
ziti edge create config odoo-host.v1 host.v1 '{"protocol":"tcp", "address":"odoo","port":8069}'

ziti edge create config odoo-intercept.v1 intercept.v1 '{"protocols":["tcp"],"addresses":["odoo.ziti"],"portRanges":[{"low":8069, "high":8069}]}'
```

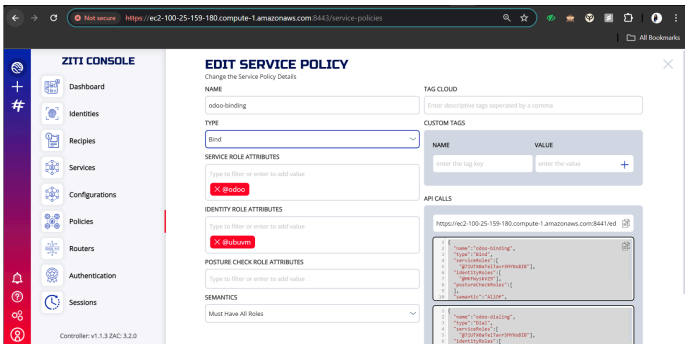
Having the configurations ready we create our service, Odoo and associate with the particular configurations from above. Therefore, service tells the edge router how to handle connections to Odoo Service.

```
ziti edge create service odoo --configs “odoo-intercept.v1” , “odoo-host.v1”
```

Managing the access to this service created we should also provide configurations
for the service access policies. We create two access policies: one that will be used to bind the service Odoo to the Ziti network overlay which will be done from the identity ubuvm that we created before and the other policy that specifies which identity will be allowed to initiate connections to the Odoo service.

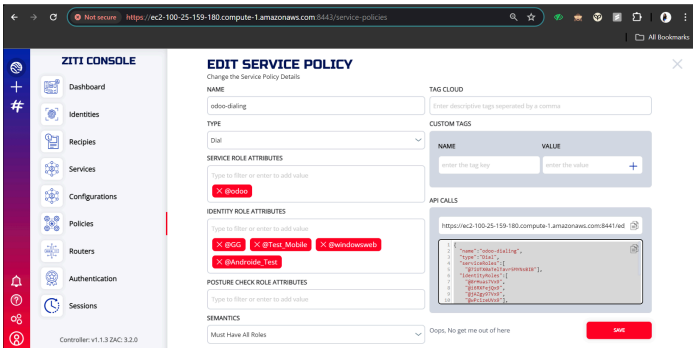
```
ziti edge create service-policy odoo-binding Bind --service-roles ‘@odoo’ --identity-roles ‘@ubuvm’
```

```
ziti edge create service-policy odoo-dialing Dial --service-roles ‘@odoo’ --identity-roles ‘@window sweb’
```



8. Bind Service Policy

End here we can see them created in the interface. For the dial service policy we have added some other identity users, created from the ZAC console that will have access to dial to the Odoo Service.

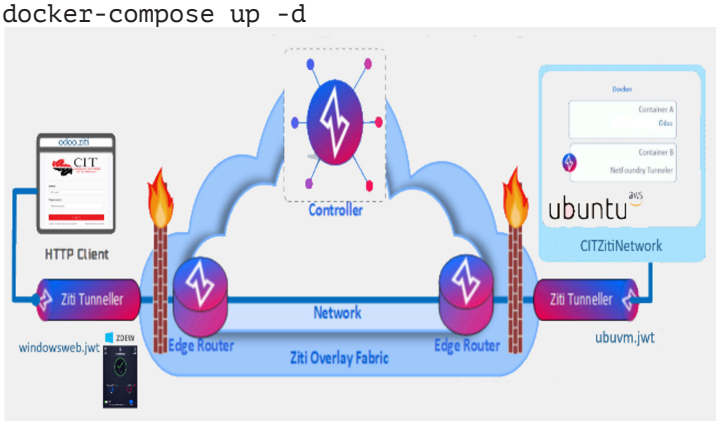


9. Dial Service Policy

2.4 Server Odoo Set Up

In this implementation step we continue with the creation of the real service that will be connected with OpenZiti. In our case, I decided to set up Odoo which is an open-source ERP that in recent years have gained popularity, especially here in Albania and is being used by most of the companies for internal management. However, every organization can follow these steps for whatever service they want to ‘zitify’, in our case a web-application.
Firstly, we create a virtual network that will serve as a bridge communication between the ziti-edge-tunneller and the web application. Then, we prepare a configuration file docker-compose which contains the images of Odoo:17 and Postgres:15 that are needed to set up this web-application.

```
docker network create CITZiti Network
docker-compose up -d
```



10. Architecture of Communication (Adapted from Openziti Official Documention Website)

When the container is created, Docker creates as a default network the one with the same name as the container. We disconnect from that default network and connect to the network we created, to make sure that Odoo container communicates only through this specified network CITZitiNetwork.

```
docker network disconnect default odoo
```

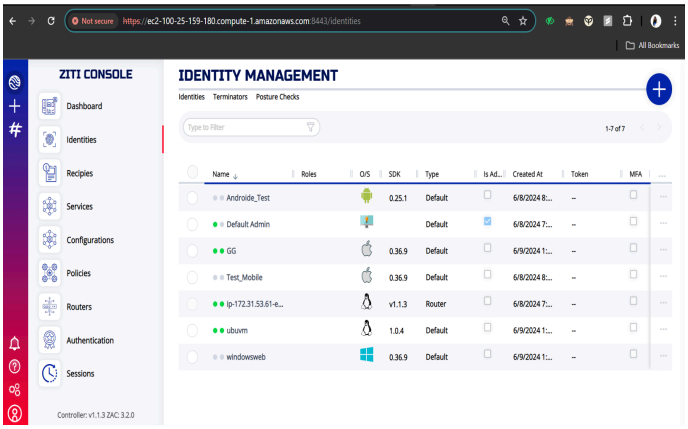
```
docker network connect CITZitiNetwork odoo
```

Continuing, it is time to create and install the docker container for the tunneller inside the web server and bind it to the ubuvm identity that we have created before. This will be as an offload point for the traffic that will be delivered to our Odoo web-application. We will create it in the same network as the Odoo containter in order to communicate through that overlay.

```
docker run \
  --rm \
  --network CITZitiNetwork \
  --network-alias zet \
  --name ziti-edge-tunnel \
  --volume ~/.ziti/ids:/ziti-edge-tunnel \
  --env=NF_REG_NAME=ubuvm \
  openziti/ziti-edge-tunnel:latest run-host
```

2.5 Identities Enrollment

For testing the communication between client and server through the Ziti Overlay Network, we have created 3 different identities. One will be tested with the Ziti Mobile Edge in IOS Phone, another in Android Phone and one in Windows with Ziti Desktop Edge. These three tunnellers will enable each of the identities enrollment and according to the particular configurations will be defined if they will access the service or not. We firstly have the windowsweb identity that we created before from the command line and assigned policies, giving access for the service Odoo. We access the token from the identity dashboard and download the One Time Token through which this identity can be enrolled and upload it in the ZDEW where we successfully enroll it. This Json Web Token has expiration and is not available for much time, so you should enroll this identity during the defined timestamp. It is for one time only so you make sure that no one else uses that for enrollment even if it has gained access to.

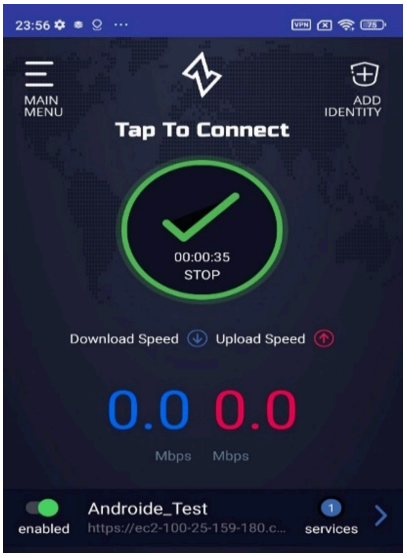


11. Identities Dashboard

If we decide at any point to forget this identity, and try to enroll again with the same JWT it will fail, because one identity is only for one device and for one time enrollment. No one else can use your identity this way minimizing the attack surface for unidentified users to access the service. Then, if any unauthorized user has access to your device and can connect through Ziti Desktop Edge, it is part of log monitoring to identify this occurrence through user behavior analysis for any anomaly activity throughout the service. However, even if this happens, the way Ziti Overlay Network is built upon, they have no chance to access other parts of network and make any kind of lateral movement, except from the applications that they have access to. Only those ports are open for that identity. This is why each identity is configured in that way with the least privileges as possible just in time and just enough access as he needs to do his tasks. When you may have identified an anomaly from that identity client, you can just remove the access policy for the services. So, the damage is minimized immediately.

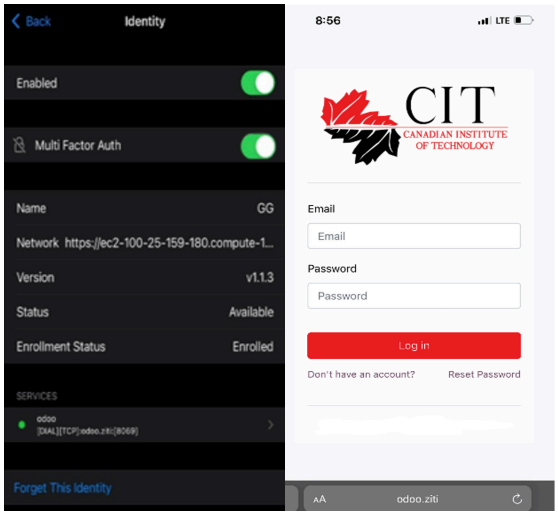
12. Identity Enrollment with QR Code

For the other use cases it will be easier to do the enrollment via QR code and scanning from the phone.



13. Successful Enrollment of Androide_Test identity Android Device

The third use case is the identity enrollment from the IOs Device. In terms of our architecture the process starts from the identity GG that dials the odoo.ziti website at port 8069. In the overlay network we have the ziti tunneller that intercepts this request and send it to the Ziti Overlay Fabric. Here the identity is authenticated and checked for access policies towards that service that has requested upon. Then, the intercept packet is delivered to the target identity, which is binded to the web-server where Odoo service is hosted.



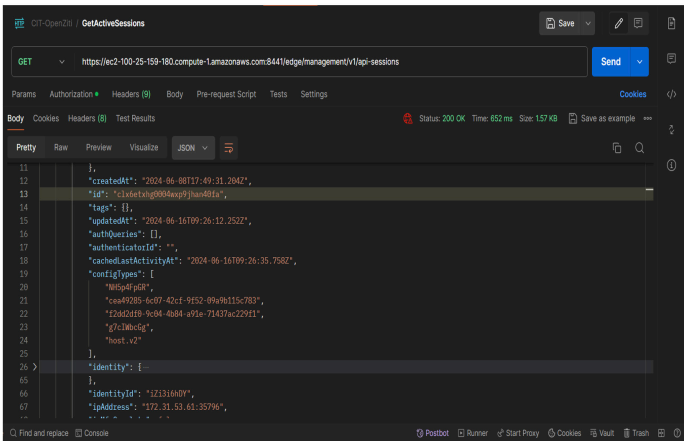
14. Successful Enrollment of GG Identity from IOs Device and access to the Odoo Service

2.6 Log Collection

Identifying the possibilities of collecting logs from the OpenZiti, we have controller logs and router logs that are real time processed. However, the information I need for possibly detecting any suspicious activity, analysing the user behavior and threat analysis, would be to take information on the active sessions in real-time. This would be better to analyse as we have the time this session was created, was updated as well as the IP address coming from. For collecting these logs, since they are not saved in any specific path inside the ziti directory, we need to retrieve them by using some API calls on the specific endpoints to get infromation on the active sessions. We started by building up a Postman Collection in order to be able firstly to retrieve these data. The specific endpoint we need is this:

```
{public-url}/edge/management/v1/
api-sessions
```

While building the right call to the endpoint, we should make sure to put the authentication method that in our case should be an API Key that is the token from the zitiLogin command in the controller where we login from the CLI. It should have the key as zt-session and the value of token. After specifying this parameter that should be added in the header we now send a Get Request to the specific endpoint getting in real time the active sessions and all the information generated for each specific one. Here is depicted as well in the image the postman request we made and the response body we got. As we can see from the response, we have all the information when a session is created, updated, when was the last activity, the IP address connected from and other specific information that we can use to detect any anomaly in these sessions and possibly track any threat. This request needs to be executed in real time in order to get the latest information on session creation.



15. Postman Request for Getting Active Sessions

Therefore, we created a scheduled function in python to run every minute in order to get the latest active sessions. This function consists of an API call, built using python libraries. The issue here for us as developers, would be the token of authorization to execute this API call because it is not always the same but it changes during a short timeframe, whereas per security it is an added value to what is pointed. We will create also an automated process to get this token, where the way this token is retrieved is by executing two commands in CLI, one for setting up the environment with all needed variables and one for login to ziti which is the command that provides the token for us to use. In order to achieve the desired results we will have an executable file that will be created in the directory as our python code. This executable file 'get_ziti_token.sh' will contain these lines of code:

```
#!/bin/bash
~/.ziti/quickstart/$(hostname -s)/$(hostname
-s).env
zitiLogin
```

The python code we build will go and execute this file through the subprocess library. And take the output that is the Token. The request that is sent by postman, we will build in python in order to execute it automatically and get the result data continuously to check for threats in real time. This function uses the library request of python to send a GET request to the specified URL and with a parameter for the token that will be provided from the get_token function specified above.

This function returns the JSON data same as in the postman response body that we will then preprocess to fetch only the data that we need.

```python
def get_token():
    try:
        # Run the shell script to source the environment and get the token
        result = subprocess.run(['./get_ziti_token.sh'], capture_output=True,
                                text=True, shell=True, executable='/bin/bash')
        token_line = [line for line in result.stdout.split('\n') if 'Token:' in line]
        if token_line:
            token = token_line[0].split('Token: ')[1]
            return token.strip()
        else:
            print(f"Error in getting token: {result.stderr}")
            return None
    except Exception as err:
        print(f"An error occurred while retrieving token: {err}")
        return None
```

16. Function to get the dynamically generated token in order to execute API

Here a suppress for the Insecure is added just in our case where we have a self-signed certificate because for demonstration purposes we have not bought a domain and applied a certificate from a certificate authority, that's why in the function we have also verify False.

```python
import subprocess
import requests
import urllib3

URL = "https://ec2-100-25-159-180.compute-1.amazonaws.com:8441/edge/management/v1/api-sessions"
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_active_sessions(token):
    headers = {
        "Content-Type": "application/json",
        "zt-session": token
    }
    try:
        response = requests.get(URL, headers=headers, verify=False)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.HTTPError as http_err:
        print(f"HTTP error occurred: {http_err}")
    except Exception as err:
        print(f"An error occurred: {err}")
        return None
```

17. API Call for collecting the information from the active sessions

2.7 Log Preprocessing

In this implementation phase, we focus on extracting the data that we are interested for in order to do analysis over them and convert into a more readable and easier way to manipulate. For this we will use the pandas library to convert our json data into DataFrames that are easier to read and manipulate. As it is depicted in the code these two functions use the pandas library that is pd and takes all the necessary data from the JSON response offered by the API, and converts them into DataFrame. The 'add_features' function is part of the Feature Engineering where we manipulate those data as we wish for having them. In our case, we convert into datetime all the objects representing a time data as well as adds some information helpful, for example we need the hour of accessing the Ziti Overlay Network and creating a session despite the date, as well as the country where the IP is located to.

```python
def parse_sessions(data):
    records = []
    for session in data['data']:
        record = {
            'createdAt': session['createdAt'],
            'updatedAt': session['updatedAt'],
            'cachedLastActivityAt': session['cachedLastActivityAt'],
            'identityId': session['identityId'],
            'ipAddress': session['ipAddress'].split(':')[0],
            'name': session['identity']['name']
        }
        records.append(record)
    return pd.DataFrame(records)


def add_features(df):
    df['createdAt'] = pd.to_datetime(df['createdAt'])
    df['updatedAt'] = pd.to_datetime(df['updatedAt'])
    df['cachedLastActivityAt'] = pd.to_datetime(df['cachedLastActivityAt'])
    df['hour'] = df['createdAt'].dt.hour
    df['country_name'] = df['ipAddress'].apply(get_country)
    df['country'] = pd.factorize(df['country_name'])[0]
    return df
```

18. Function to process the data for extracting the information we need

We need to apply another function for this and we will integrate a communication with IPInfo that through a token that we will have, we can integrate with their system for accessing the geolocation of the IP Address that we will provide from the processed data. Below is the function used to get the country name in this platform of checking the geolocation.

```python
# Initialize IPinfo handler
ipinfo_handler = ipinfo.getHandler(iptoken)

def get_country(ip):
    try:
        details = ipinfo_handler.getDetails(ip)
        return details.country_name
    except:
        return 'Unknown'
```

19. Retrieving information on the country where the IP is located

2.8 Model Training

For training a model we will use data registered from previous days or months, as longer as it is the timestamp the more data for building up a model would be and the more accurate the model will result for our anomaly detection. Firstly we prepare these data that will be in a json format and will serve as a parameter for processing these data and extracting what is needed.

```python
# Preprocess historical data
historical_data = preprocess_data(df)

# Convert categorical country data to numerical data
historical_data['country'] = pd.factorize(historical_data['country'])[0]

# Features for model training
features = historical_data[['hour', 'country']]

# Train Isolation Forest model
baseline_model = IsolationForest(contamination=0.1)
baseline_model.fit(features)

# Save the model for later use
with open('baseline_model.pkl', 'wb') as file:
    pickle.dump(baseline_model, file)

print("Baseline model trained and saved successfully.")
```

20. Model Training and Baseline Creation

We convert the country data that is categorial into numerical one by factorizing using pandas and then we create an Isolation Forest Model with a contamination rate of 10%, which is the proportion of outliers in the data that is converted into DataFrames. With the fit method of Isolation Forest we train the model using the features DataFrame. This model learns the patterns of a normal behavior based on two features, particulary with the hour of session creation and the country logged in.
We save this trained model in a file, in write-binary mode and put the serialized object of the baseline_ model in order to use it for prediction of threats.

2.9 Detecting anomalies & alerting

Detecting anomalies will be done by comparing all the data retrieved from the API call that is executed every 10 minutes with the baseline model created before.

```python
def job():
    print("Fetching active sessions...")
    token = get_token()
    if token:
        data = get_active_sessions(token)
        if data:
            df = parse_sessions(data)
            df = add_features(df)
            with open('baseline_model.pkl', 'rb') as file:
                model = pickle.load(file)
            anomalies = detect_anomalies(df, model)
            if anomalies.empty:
                print("No anomalies detected.")
            else:
                for index, row in anomalies.iterrows():
                    message = (f"Anomaly detected:\n"f"IP Address: {row['ipAddress']}\n"
                               f"Identity: {row['name']}\n"f"Country: {row['country_name']}\n"
                               f"Hour: {row['hour']}\n"f"CreatedAt: {row['createdAt']}\n"
                               f"UpdatedAt: {row['updatedAt']}\n"f"IdentityId: {row['identityId']}"
                               f"CachedLastActivityAt: {row['cachedLastActivityAt']}\n")
                    disable_identity(row['identityId'], 3, token)
                    send_slack_message(message, slack_webhook_url)
                print("Anomalies detected:", anomalies)
```

21. Job to be executed with all processes

Here we have the job that will be executed where as a flow of processes we have to retrieve the active sessions from the API with the dynamic token, preprocess these data, add manipulations to make as we need them to be structured and displayed and then load and read the binary file of the baseline model. This will be provided as a parameter for the detection of anomalies function, which does a prediction on the anomalies based on the two features that we extracted and marks with -1 the anomaly data. We iterate through these anomalies if any is found and then we send messages in a slack channel using a webhook and API for this post request. This way whoever is part of that channel is notified and alerted for this suspicious session that is created.

```python
def detect_anomalies(df, model):
    features = df[['hour', 'country']]
    df['anomaly'] = model.predict(features)
    return df[df['anomaly'] == -1]


def send_slack_message(message, webhook_url):
    headers = {'Content-Type': 'application/json'}
    data = {'text': message}
    response = requests.post(webhook_url, headers=headers, json=data)
    if response.status_code != 200:
        raise ValueError(f"Request to Slack returned an error {response.status_code}, the response is:\n{response.text}")
```

22. Anomaly Detection Function and Slack Alerts API

2.10 Response and Mitigation

For reducing the threat that might be from the anomalies that we pointed out, we bring a function built up over another API that will automate the proces of disabling the identities for a short period of time. We judged this optimal time to be 30 minutes in order for the IT team to identify if the alarm is really something to worry about, having all the information about the identity and session in real time. This is the endpoint where we have the id of the identity with anomaly behavior and we can use it as a parameter together with the minutes it needs to be disabled.

```python
def disable_identity(identity_id, duration_minutes, token):
    url = f"https://ec2-100-25-159-180.compute-1.amazonaws.com:8441/edge/management/v1/identities/{identity_id}/disable"
    headers = {
        "Content-Type": "application/json",
        "zt-session": token
    }
    data = {"durationMinutes": duration_minutes}
    response = requests.post(url, headers=headers, json=data, verify=False)
    if response.status_code == 200:
        print(f"Identity {identity_id} disabled for {duration_minutes} minutes.")
    else:
        print(f"Failed to disable identity {identity_id}. Response: {response.status_code}, {response.text}")
```

23. Disabling Identity Temporarily

2.11 Running and Testing

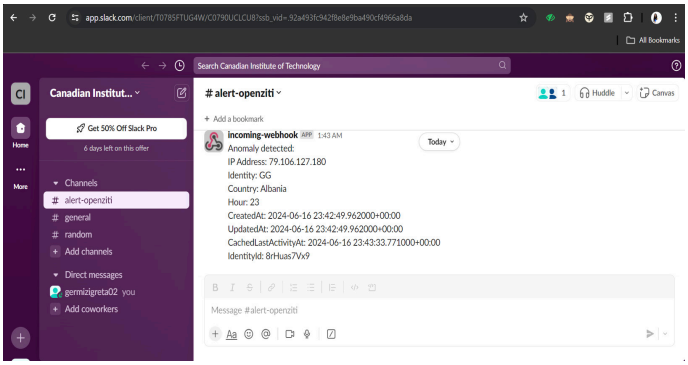
For testing purposes and demonstration we took some random data assuming employees will have their session create hour at least at 8 AM and at most 6 PM (18). As well as, we took some IP addresseswhere the country is different from Albania in order for us to understand that when we enter at 8 PM (20) and have an IP address originating from Albania we are considered as an anomaly compared to the normal data on the baseline model that is created with the random data.

```
Run:  anomaly_detect
    2 2024-06-16 22:18:24.611000+00:00  ...       1
    3 2024-06-16 22:46:15.575000+00:00  ...       1
    4 2024-06-16 22:50:16.783000+00:00  ...       1
    5 2024-06-16 23:32:10.476000+00:00  ...       0

    [6 rows x 9 columns]
    Identity i6RXFejQx9 disabled for 3 minutes.
    Identity Zo6ZVhzwFa disabled for 3 minutes.
    Identity 8rHuas7Vx9 disabled for 3 minutes.
    Anomalies detected:                          createdAt  ... anomaly
    2 2024-06-16 22:18:24.611000+00:00  ...      -1
    3 2024-06-16 22:46:15.575000+00:00  ...      -1
    4 2024-06-16 22:50:16.783000+00:00  ...      -1
```

24. Anomaly Detection Running and Results

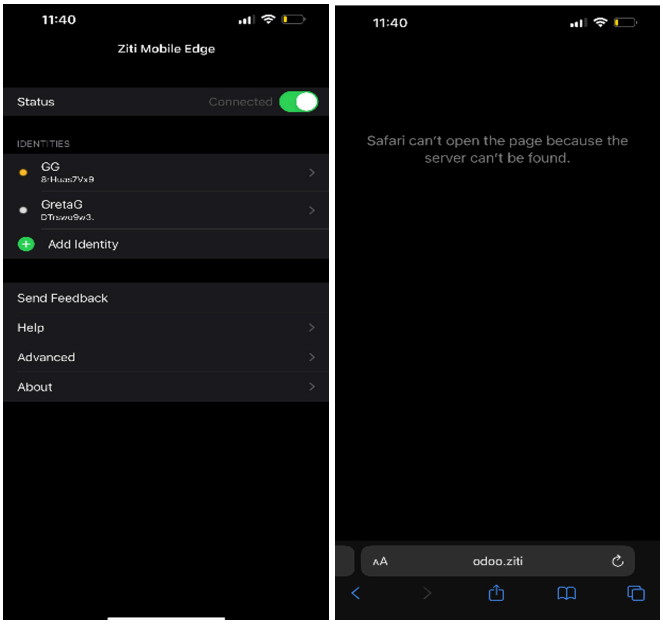
Here we can see the simulation with three identities with IP ranges originating from Albania that have created a session at 10 PM (22) and are disabled for a range of minutes that is predefined in this testing case for 3 minutes in order to try accessing again after this timestamp. The alert also is in real time where some notification comes when this session created is considered as an anomaly. The identity GG that was depicted as an



25. Slack Channel Alert Notification

anomaly is disabled and cannot access any service related even the policies defined are configured in that way to access the service. Here comes the role of the IT that can take measures towards this identity by removing access policies, or just accepting this as normal behavior and adding to the baseline model by updating the training data and retraining once again to account for these new normal behaviors. This could be done by a cron job scheduled for retraining the model with the appended data.

Here as it is shown in the identity the yellow dot means that is disabled temporarily and this GG identity cannot access the Odoo Service.



26. Disabled Identity No Access to Service

RESULTS

Implying all the security measures that are applicable to Zero Trust Network Architecture, this implementation roadmap using only open-source tools proved the effectiveness as well as the practicality for any kind of organization to enhance the security posture with this robust framework. By using open-source and vendor-neutral tools, we avoided vendor lock-in, allowing flexibility in tool selection and integration with other technologies. The modular approach used, demonstrated the flexibility for adoption in the scale the organization wants to, facilitating the process with automated deployment. Throughout this implementation, there are observed several advantages that highlight the importance of embracing ZTNA principles in today's cyberthreat landscape. With the implementation of a micro-segmented network architecture and the application of granular access policies we revealed the importance of limiting lateral movement through the network in order for the identity user not to access any other part that is not supposed to. The proposed solution to integrate User Behavior Analysis by using machine learning algorithms over Zero Trust Architecture proved to be a necessity in order to monitor in real-time any anomaly detected in the normal baseline profile that the everyday user has for any feature we define as crucial. By configuring alerts for any detected anomaly in a channel of communication, that could be easily modified in any other platform the specific organization uses for daily communications, we managed to reduce the operational burden on IT and security teams while deactivating temporarily the identity accesses until a second confirmation. This way we achieved response mechanisms and simplified management.

In summary, by embracing the Zero Trust Network Architecture with open-source tools, organizations can achieve a scalable, cost-effective, and robust security framework. This implementation demonstrated that comprehensive security measures were not only easily applicable but also essential for defending against modern cyber threats, ensuring that even smaller organizations could enhance their security posture without significant financial constraints. The practical outcomes and demonstrated benefits underscored the critical need for adopting ZTNA in today's dynamic network environments.

DISCUSSIONS

While these results affirm the value of ZTNA, certain limitations remain. Specifically, the scalability of machine learning models in high-traffic, real-time environments could present challenges for organizations with larger or more complex network infrastructures. As the volume of data increases, so do the computational demands, potentially affecting real-time detection capabilities. Additionally, implementing ZTNA across complex network architectures may require further adjustments or optimizations to ensure consistent and efficient performance across diverse use cases.

Future research could address these challenges by exploring more advanced or specialized machine learning models capable of handling the scale and diversity of large enterprise environments. Investigating alternative anomaly detection techniques such as ensemble learning or hybrid approaches may also enhance detection accuracy and reduce false positives. Continuous policy refinement, use case testing, and the addition of security layers in authentication are critical for adjusting to changing threats. Establishing long-term monitoring and assessment procedures will guarantee that security measures remain effective and resilient. Comparative studies between ZTNA and other cybersecurity frameworks could provide additional insights into ZTNA's unique strengths and potential limitations in various network settings.

CONCLUSIONS

This study demonstrates the practicality and effectiveness of implementing a Zero Trust Network Architecture (ZTNA) framework using open-source, vendor-neutral tools, proving that robust cybersecurity measures are accessible to organizations of any size. Our approach showcases how ZTNA, when integrated with machine learning-driven User Behavior Analysis (UBA), enhances real-time threat detection and response capabilities. By incorporating a micro-segmented network architecture and granular access policies, this implementation effectively minimizes lateral movement within the network, reducing the risk of data breaches and unauthorized access. This research, therefore, highlights the potential of open-source ZTNA as a flexible, scalable, and cost-effective cybersecurity solution that supports organizational resilience in an evolving threat landscape.

Key outcomes from this implementation include substantial reductions in the attack surface, real-time monitoring of user activity, and automated alerts and response mechanisms for anomalies, which collectively ease operational demands on security teams. The modularity of this open-source ZTNA approach offers scalability, enabling organizations to tailor the framework to their specific requirements and resources.

By addressing insider threats—a growing concern that accounts for a significant portion of data breaches—this study positions ZTNA as a valuable approach for protecting sensitive data without the need for proprietary solutions, which often impose significant financial and logistical constraints.

Despite these positive outcomes, our research identifies some challenges that warrant further exploration. The scalability of real-time anomaly detection in high-traffic environments, for instance, presents limitations as the volume of data increases, affecting processing capabilities and potentially introducing latency. Future work could focus on optimizing these models to handle large-scale data more effectively or exploring hybrid detection methods, such as ensemble learning, to enhance detection accuracy while reducing false positives.

Additionally, continuous policy refinement remains essential to ensure that ZTNA frameworks adapt to evolving threat landscapes. Future research should consider long-term monitoring strategies, automated policy adjustments, and advanced authentication layers to enhance resilience against sophisticated attacks. Comparative studies with alternative cybersecurity frameworks would also provide valuable insights into the unique benefits and limitations of ZTNA, further informing best practices for diverse network environments.

In conclusion, this study underscores the value of open-source ZTNA as a scalable, adaptable, and affordable solution for modern cybersecurity challenges. By blending Zero Trust principles with machine learning-driven behavioral analytics, this research contributes to the broader discourse on cybersecurity frameworks and highlights an accessible pathway for organizations to achieve high levels of security without excessive costs. Our findings demonstrate that robust, adaptable security measures can be deployed universally, ensuring that organizations remain resilient against today's complex and evolving cyber threats.

REFERENCES

Agrawal, B. W. and S. (2023, July 21). Returning to the office: The current, preferred and future state of remote work.https://www.gallup.com/workplace/397751/returning-office-current-preferred-future-state-remote-work.aspx

Annual Data Exposure Report 2023. (n.d.). Code42. Retrieved June 8, 2024, from https://www.code42.com/resources/reports/2023-data-exposure#main-content

Bispham, Mary, Creese, Sadie, Dutton, William H., Esteve-González, Patricia, & Goldsmith, Michael. (2022). An Exploratory Study of Cybersecurity in Working from Home: Problem or Enabler? Journal of Information Policy, 12, 353–386. https://doi.org/10.5325/jinfopoli.12.2022.0010

Edo, Onome, Tenebe, Imokhai, Etu, Egbe-Etu, Ayuwu, Atamgbo, Emakhu, Joshua, & Adebiyi, Shakiru. (2022). Zero Trust Architecture: Trend and Impact on Information Security. International Journal of Emerging Technology and Advanced Engineering, 12, 140-147. https://doi.org /10.46338/ijetae0722_15

Freter, R. Department of Defense. (2022, June). Department of Defense Zero Trust Reference Architecture. https://dodcio.defense.gov/Portals/0/Docum ents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf

How to mitigate insider threats by integrating UEBA with Zero trust - ManageEngine. (n.d.). https://www.manageengine.com/log-management/ebooks/integrating-ueba-with-zero-trust-to-secure-business.html

Implementing a Zero trust security model at Microsoft. (2024, March 12). Inside Track Blog. https://www.microsoft.com/insidetrack/blog/implementing-a-zero-trust-security-model-at-microsoft/

Loukides, M. (2021, December 7). The cloud in 2021: Adoption continues. O'Reilly Media. https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/

Nicoletti, Pete. (2022, April 19). Remote work security statistics in 2022 - CyberTalk. CyberTalk. https://www.cybertalk.org/2022/03/31/remote-work-security-statistics-in-2022/

No More Chewy Centers: The Zero Trust Model Of Information Security | Forrester. (n.d.). Forrester. https://www.forrester.com/report/No-More-Chewy-Centers-The-Zero-Trust-Model-Of-Information-Security/RES56682

NIST. (n.d.). Embracing a zero-trust security model. https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF

Ponemon Cost of Insider Threats Global Report | Proofpoint US. (2024, May 10). Proofpoint. https://www.proofpoint.com/us/resources/threat-reports/cost-of-insider-threats

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2019). Zero trust architecture (NIST 800-207). NIST, Gaithersburg, MD, USA.

Spencer, Matt, & Pizio, Daniele. (2023). The de-perimeterization of information security: The Jericho Forum, zero trust, and narrativity. Social Studies of Science. https://doi.org/10.1177/03063127231221107

Twilio. Covid-19 Digital Engagement Report. (n.d.). https://pages.twilio.com/rs/294-TKB-300/images/Twilios-Covid-19-Digital_Engagement_Report_4832.pdf

Ward, R., & Beyer, B. (2014, December). BeyondCorp: A new approach to enterprise security. Google Research. https://research.google/pubs/beyondcorp-a-new-approach-to-enterprise-security/

Zippia. "26 Surprising BYOD Statistics [2023]: BYOD Trends In The Workplace." Zippia.com. Oct. 17, 2022, https://www.zippia.com/advice/byod-statistics

Himanshu Sharma. BEHAVIORAL ANALYTICS AND ZERO TRUST. International Journal of Computer Engineering and Technology, 2021, 12 (1), pp.63-84. ffhal-04686453

Zhou, X., & Du, J. (2021). Leveraging AI for enhanced threat detection in zero trust environments: A systematic review. Cybersecurity, 4(2), 1-15. https://doi.org/10.1186/s42400-021-00085-3