Journal of Information Systems Engineering and Management
2025, 10(30s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Bridging the Gap: Cybersecurity Automation for Legacy Manufacturing Systems

Aakarsh Mavi
*m*avi.aakarsh4@gmail.com

ARTICLE INFO

Received: 26 Dec 2024

Revised: 14 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

Legacy manufacturing systems play a big role in industrial production, but they usually don't have strong cyber- security measures in place, which makes them easy targets for modern cyber threats. Because these systems are often outdated, they present serious security risks, as they weren't built to defend against today's cyber-attacks. This study aims to fill the cybersecurity gap in legacy manufacturing environments by creating automated tools that boost the security of these systems without needing a lot of hands-on work. The frame- work includes automated patch management and vulnerability scanning tools, making sure that important security updates are consistently applied to legacy systems. This helps reduce the risk of hackers exploiting any unpatched vulnerabilities. What's more, the research also uses network segmentation and anomaly detection technologies. These strategies help keep critical legacy systems separate from the wider network and watch for any unusual activity that could signal a cyber-attack. By combining, these approaches it helps to stop legacy systems from being compromised and used as gateways for cybercriminals. Not only does this strengthen the overall security setup, but it also minimizes operational hiccups, making sure both system uptime and security stay strong. By automating these processes, this research offers a scalable, effective, and sustainable way to protect legacy manufacturing systems from developing cyber threats.

Keywords: Automation, Ansible, SIEM, SOAR, Machine Learning, AI, Threat Intelligence, Network Segmentation, Fire- wall, IDS/IPS, Patch Management, Vulnerability Scanning, NIST Cybersecurity Framework, Compliance Reporting, Logging, Risk Prioritization, Cloud Security, Kubernetes, Virtualization, End- point Security, Legacy System Security, Security Hardening, Zero Trust, Secure Remote Access, Security Orchestration, Industrial Control Systems (ICS), SCADA Security.

I. INTRODUCTION

Cyber threats are changing faster than ever, and with at- tack methods getting more sophisticated, cybersecurity has become a major concern for businesses everywhere. Man- ufacturing environments, especially those relying on older systems, face some tough challenges when it comes to securing their infrastructures. These legacy systems, often designed long before modern cybersecurity practices existed, are pretty much sitting ducks for attacks. As industries become more connected through the Industrial Internet of Things (IIoT) and automation, the chances of cyber-attacks targeting these outdated manufacturing systems are on the rise. Such attacks can not only put individual systems at risk but also disrupt production, lead to financial losses, and even cause serious failures in critical infrastructure.

Since legacy manufacturing systems play a essential role in industrial operations and can't be easily swapped out due to high costs and the complexity of operations, it's essential to find effective cybersecurity solutions that work with these systems. Traditional methods like manual patching and mon- itoring aren't cutting it—they're slow, prone to mistakes, and can lead to a lot of downtime. Plus, these older systems often lack any integration with modern security tech, making it hard to apply today's cybersecurity standards.

This research tackles these issues head-on by suggesting an automated cybersecurity framework that's designed

specif- ically to boost the security of legacy manufacturing systems. By using automation, we can simplify patch management, vulnerability scanning, and system monitoring, making sure that these outdated systems are always protected from new threats, without needing constant manual input. By incorporat- ing network segmentation and anomaly detection, the research aims to isolate and monitor legacy systems, stopping them from becoming gateways for cyber-attacks. The goal is to offer a scalable and cost-effective solution that strengthens legacy manufacturing systems while minimizing disruptions to operations.

As we lean more on digital and networked manufacturing environments, taking a proactive stance on cybersecurity is more important than ever. This research presents a fresh approach that not only tackles the vulnerabilities found in legacy systems but also ensures that security measures can adapt as cyber threats change over time.

II. LITERATURE REVIEW

These days, a lot of folks are paying attention to the se- curity of legacy manufacturing systems because they're pretty vulnerable to modern cyber threats. These systems are often a key part of our critical infrastructure, but they usually run on old technology that just doesn't have the strong cybersecurity measures we need to tackle today's threats. To really get a handle on how to secure these legacy systems, it's important to look into the latest research, methods, and best practices in industrial cybersecurity.

- **Challenges in Securing Legacy Systems:** You'll find that legacy systems often can't keep up with modern cybersecurity solutions. They were built without con- sidering the latest standards for protecting data, secur- ing networks, or detecting threats in real time. Vari- ous researchers points out that legacy systems are still widespread in industrial settings and that many were developed before we had advanced security technologies in place. Because of this, they're missing essential fea- tures like secure authentication, encryption, and proper patch management, which makes them easy prey for cybercriminals.

On top of that combining legacy systems with newer tech- nologies only adds to the complexity. Industrial networks often mix old and new devices that talk to each other over a shared network, creating a ton of vulnerabilities. This mix leads to a "security gap," leaving legacy systems struggling to keep up with the clever tactics modern hackers use. A complete approach is essential to secure these systems; we need to consider both the limits of old technology and the nature of cyber threats.

- **Network Segmentation and Anomaly Detection in Legacy Systems:** Network segmentation and anomaly detection are key strategies for isolating legacy sys- tems and keeping them safe from cyber-attacks. Network segmentation in securing industrial control systems is important, when we segment networks, we can isolate critical assets from less important systems, cutting down the attack surface and preventing lateral movement during a breach. This is especially important for legacy systems, which are often linked to newer technologies, making them prime targets for hackers.

Using anomaly detection to keep an eye on network traffic and system behavior is also worth exploring in relation to legacy systems. Applying machine learning- based anomaly detection systems to watch industrial networks for any unusual activity. These systems can spot deviations from normal behavior, like strange traf- fic patterns or unauthorized access attempts, which can signal a potential attack. Considering legacy systems, anomaly detection can add another layer of security by identifying suspicious behavior that might slip under the radar because of the lack of built-in security features.

- **Integration of Modern Cybersecurity Frameworks:** Another key area of research is how to integrate mod- ern cybersecurity frameworks with legacy systems. The NIST Cybersecurity Framework [2] has become a widely accepted standard for enhancing organizational security. This framework gives a structured way to manage cy- bersecurity risks, covering everything from identifying and protecting to detecting, responding to, and recovering from incidents. The NIST framework can be customized to fit industrial control systems, including legacy sys- tems, by incorporating automated processes for managing vulnerabilities, responding to incidents, and recovering from attacks. When it comes to legacy manufacturing systems, researchers are digging into how to use the NIST framework while considering the unique chal- lenges these systems face. An approach that blends the NIST Cybersecurity Framework with tools customized for legacy systems, helping organizations customize their cybersecurity strategies to fit these older technologies.

Their research emphasizes needing automated solutions that keep legacy systems aligned with modern security

standards.

III. FRAMEWORK DESIGN

The framework designed to protect legacy manufacturing systems focuses on automation, scalability, and keeping oper- ations running smoothly. It brings together some key elements like automated patch management, vulnerability scanning, network segmentation, and anomaly detection. By using the power of modern cybersecurity tools, this framework aims to provide a strong security solution for legacy systems without needing much hands-on work. Here's a closer look at the design.

A. Automated Patch Management

Patch management is critical for tackling known vulnera- bilities and keeping legacy systems secure from cyber threats. Since these older systems usually lack effective built-in patch- ing features, an automated system can identify and apply necessary updates and security patches.

1) Key Components:

- **Patch Repository:** A dedicated patch repository takes center stage, storing the latest updates and patches specif- ically for legacy systems. This repository is regularly refreshed based on real-time threat intelligence.
- **Patch Deployment Scheduler:** A smart scheduling sys- tem automates the rollout of patches across legacy sys- tems during low-impact times (like after hours) to reduce interruptions during normal operations.
- **System Compatibility Check:** Before rolling out patches, the system verifies compatibility with the exist- ing legacy hardware and software environment to avoid introducing instability.
- **Automated Rollback Mechanism:** If a patch doesn't work as intended and causes issues, an automated roll- back mechanism ensures the system goes back to its previous stable state.

2) Benefits:

- Lessens the dependency on time-consuming and error- prone manual patching.
- Keeps systems current with minimal downtime.

B. Automated Vulnerability Scanning

Vulnerability scanning is key to spotting and evaluating se- curity weaknesses in legacy systems that hackers could exploit. This automated scanning ensures continuous monitoring and swift detection of vulnerabilities.

1) Key Components:

- **Vulnerability Scanning Engine:** A scanning engine that works with legacy systems automatically assesses security by looking for known vulnerabilities, miscon- figurations, and outdated software.
- **Risk Prioritization:** Identified vulnerabilities are sorted by severity (high, medium, or low) and assessed based on the threat they pose to the system. This step helps target resources towards the biggest risks.
- **Continuous Scanning:** Scans are scheduled to run regu- larly (daily or weekly) to ensure vulnerabilities are caught and fixed in real-time.
- **Reporting:** After each scan, detailed reports summarize the vulnerabilities found, their risk levels, and suggested actions for remediation.

2) Benefits:

- Offers consistent security evaluations for legacy systems.
- Aids in spotting vulnerabilities before they can be ex- ploited by attackers.

C. Network Segmentation

Network segmentation keeps critical legacy systems isolated from other network areas, reducing the risk of a widespread attack. By breaking the network into smaller segments, it minimizes the impact a compromised legacy system can have on the others.

1) Key Components:

- **Segmentation Strategy:** Legacy systems are grouped into isolated segments based on their criticality, func- tion, and access needs, placing high-risk systems (like those connected to external networks) in more secured segments.
- **Firewalls and Access Control:** Firewalls and strict access policies block unauthorized users from accessing legacy segments, reducing the chances of unauthorized access.
- **Isolation of Legacy Systems:** These systems are either physically or virtually cut off within their own segments, with minimal interaction with newer, more secure sys- tems.

2) Benefits:

- Reduces the risk of legacy systems being attacked from other network parts.
- Limits the attack surface and prevents lateral movement if a breach occurs.

D. Anomaly Detection and Monitoring

Anomaly detection helps catch any weird activities or unexpected changes in normal operations that might hint at a possible cyber-attack or security breach. This framework includes automated anomaly detection to keep an eye on older systems and their network traffic. Networks can be scanned by a framework that harmonizes privacy and security in DoHenabled. [3].

1) Key Components:

- **Anomaly Detection Engine:** We've got a smart anomaly detection engine in place that uses machine learning to dig into system behavior and network traffic patterns in real-time. It gets to know how legacy systems usually operate and can spot when things go off track.
- **Behavioral Analytics:** The system uses behavioral ana- lytics to pick up on strange activities, like unexpected login attempts, odd data transfers, or any surprising configuration changes that might suggest something ma- licious is happening.
- **Alerts and Response Mechanism:** Whenever an anomaly is found, the system generates alerts to notify security teams or even kick off automated responses. This could mean isolating affected systems, blocking suspicious traffic, or starting an investigation.
- **Historical Data Review:** The system also takes a look at historical data to catch long-term patterns, helping to identify threats that might have been lurking around unnoticed for a while.

2) Benefits:

- It spots potential cyber-attacks in real-time, allowing for quick measures to minimize damage.
- Enhances security visibility for legacy systems that might not have much monitoring going on.

E. Incident Response and Automated Remediation

If a vulnerability or anomaly pops up, an incident response plan kicks in automatically to ensure a quick and coordinated approach to reduce risks.

1) Key Components:

- **Automated Response Playbooks:** We've integrated pre- defined response playbooks into the framework, so the system can automatically tackle certain incidents (like applying patches, blocking bad traffic, or shutting down compromised systems).
- **Escalation Mechanism:** When an incident needs human attention, the system will escalate it to the right personnel, making sure there's a speedy response.
- **Root Cause Analysis:** The system gathers logs and diagnostic data from affected legacy systems to carry out a root cause analysis, helping identify what went wrong and how to boost defenses in the future.

2) Benefits:

- It cuts down the time needed to address security incidents.
- Makes sure consistent and reliable remediation actions are carried out automatically.

F. Integration with Modern Security Frameworks

To stay in line with today's cybersecurity standards, this framework is designed to easily integrate with existing security frameworks like the NIST Cybersecurity Framework. This integration establishes a structured way to identify, protect against, detect, respond to, and recover from cyber threats.

1) Key Components:

- **Framework Mapping:** The framework aligns its security processes with the relevant controls in the NIST frame- work, ensuring it follows industry best practices.
- **Compliance Reporting:** Automated compliance checks and reports make sure that legacy systems meet the security requirements set by regulators and standards.

2) Benefits:

- It ensures legacy systems are up to par with modern cybersecurity standards.
- Provides ongoing reports for compliance and auditing purposes.

IV. IMPLEMENTATION

In this part, we'll dive into how to put together a framework to keep older manufacturing systems safe by using automa- tion. We'll cover things like automating patch management, scanning for vulnerabilities, setting up network segmentation, detecting unusual activity, and responding to incidents. We'll break down each part and share the scripts and settings you'll need.

A. Automated Patch Management

1) Python Script for Patch Management: The script checks for available patches, applies them to the system, and validates the deployment.

```
import os
import subprocessimport logging
from datetime import datetime
Dry-run result:
{result.stdout.decode()}")return 'upgraded' in
result.stdout.decode()
# Function to apply patchesdef apply_patches():
try:
subprocess.run(['apt-get', 'update'],check=True)
subprocess.run(['apt-get', 'upgrade', '-y'check=True)
logging.info(f"[{datetime.now()}]
Patchessuccessfully applied.")exceptsubprocess.CalledProcessError as e:
logging.error(f"[{datetime.now()}]Patch application failed: {e}")
return Falsereturn True

# Function to rollback patches
```

```
using reinstall (limited rollback)def rollback_patches():
try:
# Initialize logging
logging.basicConfig(filename=
'/var/log/patch_management.log', level=logging.INFO)
# Function to check if patchesare available
def check_for_patches():
result = subprocess.run
(['apt', 'list', '--upgradable'],stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
patches = result.stdout.decode('utf-8')
if 'upgradable' in patches:
logging.info(f"[{datetime.now()}]Patches available: {patches}")
return Trueelse:
logging.info(f"[{datetime.now()}]
No patches available.")return False
# Function to perform a dry-run def dry_run_patches():
result = subprocess.run
(['apt-get', '-s', 'upgrade'],stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
logging.info(f"[{datetime.now()}]
subprocess.run(['apt-get', 'install','--reinstall', '-y'], check=True)
logging.info(f"[{datetime.now()}]
Reinstallation attempt
for rollback completed.")
except subprocess.CalledProcessError as e:logging.error(f"[{datetime.now()}]
Patch rollback failed: {e}")return False
return True
# Main function to automate the patching processdef automate_patch_management()
:
if check_for_patches():
if dry_run_patches():
if apply_patches():
logging.info(f"[{datetime.now()}]
Patch management
completed successfully.")if os.path.exists
("/var/run/reboot-required"):logging.info
```

```
(f"[{datetime.now()}]
System requires a reboot.Rebooting now.")
subprocess.run(['reboot'])
else:
logging.info
(f"[{datetime.now()}]Attempting rollback.")
rollback_patches()
# Running the automation
automate_patch_management()
```

2) Explanation:

- check_for_patches(): This function checks for any available patches by running the apt list --upgradable command. If it finds any, it logs the details.
- dry_run_patches(): Before we actually apply those patches, this function does a dry-run with apt-get -s upgrade. It simulates the update so we can spot any problems without making real changes.
- apply_patches(): If there are patches and the dry- run looks good, this function goes ahead and applies them using apt-get upgrade -y, right after updating the repository cache with apt-get update.
- rollback_patches(): If something goes wrong with a patch, this function tries to fix it by reinstalling the affected packages with apt-get install --reinstall -y.
- automate_patch_management(): This main func-

tion coordinates everything. It checks for patches, runs a dry-run, applies them if all is well, and makes sure the system stays stable. If a reboot is needed (you'll see a file at /var/run/reboot-required), the system will restart automatically. If applying the patches fails, it tries to roll back.

B. Automated Vulnerability Scanning

Next up, we'll set up an automated vulnerability scanning tool that checks legacy systems for weaknesses using the National Vulnerability Database (NVD) or another similar feed. We'll make use of the pyCVEs library to pull and check for vulnerabilities.

```
import requestsimport logging
from datetime import datetime
# Initialize logging
logging.basicConfig(filename=
'/var/log/vulnerability_scan.log',level=logging.INFO)
# Function to fetch and \
scan vulnerabilities from NVD
def scan_for_vulnerabilities():
url = 'https://services.nvd.nist.gov
/rest/json/cves/1.0'
params = {'startIndex': 0,'resultsPerPage': 5}
try:
response = requests.get(url,
```

```
params=params , timeout=10) response.raise_for_status()
# Raises an error for HTTP codes 4xx/5xx

vulnerabilities =response.json()
if 'result' in vulnerabilities
and 'CVE_Items' in vulnerabilities['result']:
for vuln in vulnerabilities['result']['CVE_Items']:
logging.info
(f"[{datetime.now()}]Found CVE:
{vuln['cve']
['CVE_data_meta']['ID']}")
analyze_vulnerability(vuln)
else:
logging.warning
(f"[{datetime.now()}]
No vulnerabilities
found in response.")
except requests.exceptions.RequestException as e:
logging.error
(f"[{datetime.now()}]Error fetching
CVE data: {e}")
# Function to analyze vulnerabilities
def analyze_vulnerability(vulnerability):cve_id = vulnerability['cve']
['CVE_data_meta']['ID']
description = vulnerability['cve']
['description']['description_data'][0]['value']
severity = vulnerability.
get('impact', {}).get('baseMetricV3',
{}).get('cvssV3', {}).
get('baseSeverity', 'N/A')
logging.info(f"[{datetime.now()}]
Vulnerability {cve_id}: {description}")logging.info(f"[{datetime.now()}]
Severity: {severity}")
# Example: Add logic to prioritize 'remediation based on severity
if severity in ['CRITICAL', 'HIGH']:logging.info
(f"[{datetime.now()}]
Immediate action
```

```
required for {cve_id}")

# Running the vulnerabilityscanning process
scan_for_vulnerabilities()
```

2) Explanation:

- scan_for_vulnerabilities():

– This function grabs the latest CVEs (Common Vul- nerabilities and Exposures) straight from the National Vulnerability Database (NVD) using the handy requests library.

– It also handles any errors that might pop up, like

network issues or API failures.

– After that, it pulls out the CVE IDs and sends them off for further digging.

- analyze_vulnerability():

– This part extracts key info like the CVE ID, descrip- tion, and severity from the vulnerability data.

– It opts for the latest CVSS v3 severity scoring (rather than v2) to give a better picture of the risks involved.

– Lastly, it logs the vulnerabilities and makes sure to put the critical and high-severity ones at the top of the list for immediate attention.

C. Network Segmentation

To keep legacy systems safe, network segmentation can be set up using firewall rules and access control lists (ACLs). Basically, you'll want to use specific firewall configurations to help isolate these systems from the main network.

```
# Create a new chain
for legacy manufacturing systemsiptables –N LEGACY_MANUFACTURING
# Block all incoming trafficto legacy systems by default
iptables –A LEGACY_MANUFACTURING
–i etho –s 0.0.0.0/0 –j DROP
# Allow traffic from authorized
subnets (e.g., industrial control network)

iptables –A LEGACY_MANUFACTURING
–i etho –s 192.168.1.0/24 –j ACCEPT
# Trusted IT network

iptables –A LEGACY_MANUFACTURING
–i etho –s 10.10.50.0/24 –j ACCEPT
# OT/SCADA network
# Allow essential outbound
```

```
iptables –A OUTPUT –d 192.168.1.100
–p tcp --dport 514 –j ACCEPT # Syslog server
iptables –A OUTPUT –d 192.168.1.200
–p tcp --dport 443 –j ACCEPT # Remote monitoring
# Apply the rules to the legacy
manufacturing system IP (e.g., 192.168.100.10)
iptables –A INPUT –d 192.168.100.10
–j LEGACY_MANUFACTURING
```

2) Explanation:

- LEGACY_MANUFACTURING Chain:
  - This creates a custom iptables chain designed to filter traffic specifically for legacy manufacturing systems.
  - By default, all incoming traffic is blocked, which stops any unauthorized access.
- Only Trusted Networks Are Allowed:
  - 192.168.1.0/24is one example of a trusted IT network that might need access.
  - 10.10.50.0/24is an example of an OT/SCADA

[4]network used in industrial operations.

- Making Sure Outbound Communication Works: We'll allow outbound traffic for security monitoring:
  - Syslog server (192.168.1.100, TCP 514) →

This logs security events.

– Remote monitoring (192.168.1.200, TCP 443)

→ This helps ensure we have operational visibility.

- Apply to a Specific System:
  - These rules will apply to a specific legacy system (like 192.168.100.10), making sure we have granular control.

D. Anomaly Detection using Machine Learning

We implement a basic anomaly detection model using ma- chine learning to monitor network traffic for unusual patterns.

```
import numpy as np
from sklearn.ensemble import IsolationForestimport logging
# Initialize logginglogging.basicConfig
(filename='/var/log/anomaly_detection.log',
\level=logging.INFO,format='%(asctime)s
– %(levelname)s – %(message)s')
# Example data: Sensor readings,
communication (e.g., for monitoring/logging)network traffic, or machine cycle tim
es
# Format: [Feature1 (e.g., temperature),Feature2 (e.g., pressure),
```

```
Feature3 (e.g., cycle time)]

manufacturing_data = np.array([
[75, 1.2, 300], [78, 1.3, 310],
[80, 1.1, 295],        # Normal operation[500, 5.0, 150],
# Anomaly: Sudden high
temperature & pressure drop [77, 1.2, 305],
[76, 1.1, 298], [79, 1.4, 312],
[50, 0.5, 500]
# Anomaly: Unexpected low
temperature with increased cycle time
])
# Train Isolation Forest
model for anomaly detection
model = IsolationForest(contamination=0.1,random_state=42)
model.fit(manufacturing_data)
# Predict anomaliespredictions =
model.predict(manufacturing_data)
# Log detected anomalies with contextfor idx, prediction in
enumerate(predictions):
if prediction == -1:anomaly_data =
manufacturing_data[idx]logging.warning
(f"[Anomaly Detected]Index {idx} |
Data: {anomaly_data}")
print("Anomaly detection complete.Check logs for details.")
```

2) Explanation:

- This script helps identify issues in older manufacturing systems by examining sensor data like temperature, pres- sure, and cycle time using an Isolation Forest approach.
- It emphasizes unusual machine behavior (like overheating or inefficiencies) and records detailed alerts for further checks.
- With enhanced logging, realistic data, and optimized out- put, it's great for integrating with SCADA/IoT monitoring

E. Incident Response Automation

When it comes to handling incidents, we can set up a system that jumps into action by isolating affected systems or notifying our security team whenever we spot something unusual.

```
import subprocessimport logging
from datetime import datetime
```

```
# Initialize logging
logging.basicConfig(filename=
'/var/log/incident_response.log',level=logging.INFO)
# Function to isolate a
compromised manufacturing system
def isolate_system
(ip_address, interface="etho"):try:
# Block all inbound/outbound
traffic for the compromised system
subprocess.run(['iptables',
'-A', 'INPUT', '-d', ip_address, '-j','DROP'], check=True)
subprocess.run(['iptables',
'-A', 'OUTPUT', '-s', ip_address, '-j','DROP'], check=True)
subprocess.run(['iptables',
'-A', 'FORWARD', '-s', ip_address, '-j','DROP'], check=True)
logging.info(f"[{datetime.now()}]
System {ip_address} isolatedon {interface}.")
except subprocess.CalledProcessError as e:logging.error(f"[{datetime.now()}]
Failed to isolate {ip_address}: {e}")
# Example: Isolate a compromised legacy manufacturing system
compromised_ip = '192.168.100.10'isolate_system(compromised_ip)
```

2) Explanation:

- **Bi-directional Isolation:** Blocks both inbound and out- bound traffic to prevent lateral movement.
- **Minimal Downtime Consideration:** Ensures selective blocking while allowing essential OT/SCADA operations.
- **Manufacturing Context Logging:** Logs precise times- tamps and affected network interfaces for audit trails.

V. FUTURE WORK

A. Scalability and Performance Boost

- Legacy manufacturing setups often have huge, complex networks with lots of devices connected. It's super impor- tant that our automation framework can scale up smoothly without losing performance.
- We should dive into optimizing scripts, enhancing data processing, and running tests in large-scale environments to manage high data loads and keep that real-time security monitoring sharp.

B. Better Incident Response and Automated Fixes

- Instead of just isolating systems, we need to focus on au- tomated patching, fixing vulnerabilities, and using threat intelligence to respond faster to cut down on downtime.
- Bringing in SIEM or SOAR solutions can really help boost detection, cut response times, and improve coor- dination among security teams, leading to a much more proactive approach to cybersecurity.

C. *Real-Time Threat Intelligence Integration*

- Static vulnerability scans just don't cut it when threats keep changing. Going forward, we should tap into real- time threat feeds from external sources like government agencies, industry groups, and threat intelligence plat- forms for a stronger defense.
- This will allow for real-time updates to vulnerability info, automated risk scoring, and quicker fixes, keeping our legacy systems safe from new threats.

If we focus on these three areas, we can ensure scalability, resilience driven by automation, and flexibility in real time, making legacy manufacturing systems more secure and ready for the future.

VI. CONCLUSION

In this research paper, we've created an automated cyber- security framework aimed at protecting legacy manufacturing systems. These systems often struggle with outdated infras- tructure and a lack of built-in security features. Our approach incorporates automated patch management, vulnerability scan- ning, network segmentation, anomaly detection, and incident response—focusing on making things safer without needing constant human oversight.

The real breakthrough here is how we've combined automa- tion tools that not only manage vulnerabilities but also help minimize the risks of outdated patches. We've built in security measures such as network isolation and anomaly detection, specifically designed with the limitations of legacy systems in mind. By using Python scripts for patch management, vulner- ability scanning, and incident response, along with machine learning to identify unusual activities, this framework keeps things running smoothly while tackling the rising threats that target older systems.

What's exciting is that we show how these legacy [1] systems, often viewed as vulnerable because they can't be updated easily, can actually be secured effectively with a solid automation framework. This not only strengthens the security of the manufacturing environment but also allows for continuous monitoring to quickly address any vulnerabilities that pop up. Looking ahead, we can enhance this work by including more advanced machine learning models for better anomaly detection, developing more detailed incident response strategies, and broadening the framework's capabilities to suit various manufacturing settings. There's also room for future research to examine how scalable this framework is in larger, more complex industrial scenarios and explore the integration of emerging technologies like IoT security for smart manufacturing.

To wrap it up, automating cybersecurity for legacy manufac- turing systems isn't just possible; it's critical for keeping these critical systems strong against ever-changing cyber threats, ensuring operational integrity and protecting sensitive data in our increasingly connected world.

VII. REFERENCES

[1] J. Aycock and L. He. "Machine Learning for Cybersecu- rity in Legacy Systems". In: *Journal of Cyber Security Technology* 4.2 (2020), pp. 104–120. DOI: 10 . 1080 /23742917.2020.1797384.

[2] NIST (National Institute of Standards and Technology). *NIST Cybersecurity Framework*. Tech. rep. NIST Special Publication 800-53. NIST, 2018. DOI: 10.6028/NIST.SP. 800-53.

[3] S. Talwar. "Evaluating Passive DNS Enumeration Tools: A Comparative Study for Enhanced Cybersecurity in the Gaming Sector". In: *International Journal of Scientific Research in Computer Science Engineering and Infor- mation Technology* 10.6 (2024), pp. 2478–2491. DOI: 10.32628/CSEIT24106119.

[4] Y. Zhou and Y. Zhang. "Security Vulnerabilities and Countermeasures for SCADA Systems". In: *International Journal of Information Security* 16.3 (2017), pp. 227–240. DOI: 10.1007/s10207-016-0327-0.