Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Comprehensive Threat Assessment and Penetration Testing of a Smart Refrigerator System

Shubham[1], Dr.Rajinder Singh Sodhi[2], Dr.Preet Kaur[3]

[1]*Research Scholar, Department of Computer Science, Om Sterling Global University, Hisar*
[2]*Supervisor, Department of Computer Science, Om Sterling Global University, Hisar*
[3]*Co-supervisor, Department of Electronics Engineering, J C Boss University of Science and Technology, YMCA Faridabad*

This paper evaluates the vulnerabilities and security threats present in a smart refrigerator system powered by an embedded operating system. By conducting a detailed threat assessment using passive and active information gathering techniques, we identify critical system assets, decompose the system for analysis, and map potential attack vectors. Using the STRIDE model, we systematically assess spoofing, tampering, information disclosure, denial of service, and privilege escalation risks. Penetration tests are performed to exploit identified vulnerabilities, testing both hardware and software components. The results emphasize the importance of security measures, while acknowledging areas for improvement in mitigating advanced attacks.

Keywords: IoT security, smart refrigerators, STRIDE model, penetration testing, embedded systems, cloud security, firmware vulnerabilities, mobile applications, identity spoofing, cyber-attacks.

1. Introduction

The Internet of Things (IoT) has revolutionized everyday appliances by embedding them with advanced digital capabilities, allowing seamless integration with mobile applications, cloud services, and other network-connected devices. Smart refrigerators are a prime example of this transformation, offering a range of features that include remote monitoring, media sharing, inventory management, and interaction with smart home ecosystems. However, this increased functionality also introduces vulnerabilities, making such devices a target for cyber-attacks. As these smart devices become more common in households, they attract attention from malicious actors seeking to exploit their potential security weaknesses. Threats like data breaches, identity spoofing, unauthorized data manipulation, and even denial-of-service (DoS) attacks are real dangers for consumers who rely on IoT devices like smart refrigerators to manage daily tasks.

This paper aims to assess these risks by performing a comprehensive security evaluation and penetration testing of a smart refrigerator system. By identifying critical assets, decomposing

the system architecture, and using well-established threat modeling techniques like STRIDE, we can pinpoint potential vulnerabilities. Additionally, the penetration testing of both hardware and software components offers insights into how attackers might exploit these weaknesses, providing recommendations for mitigating such risks.

Key objectives:

- Identify key assets of the smart refrigerator system and their associated attack vectors.
- Assess potential vulnerabilities using the STRIDE threat modeling methodology.
- Conduct penetration tests on the system to simulate potential cyber-attacks and evaluate system responses.
- Provide suggestions for improving the system's security, reducing exposure to attacks.

Smart refrigerator systems, such as the one analyzed in this paper, integrate sophisticated communication protocols and multi-layered services, which can make them complex to secure fully. While these systems improve user experience and add convenience, they also require a robust security framework to protect user data and prevent unauthorized access.

2. Background

Smart refrigerators represent an increasingly important segment of the IoT landscape, connecting everyday devices to the internet and providing interactive functionalities. These devices use embedded operating systems and rely on various cloud services, mobile applications, and internet connectivity, which expose them to numerous cybersecurity challenges (Ali, 2012; Misra & Dhir, 2012). Commonly, these devices employ Linux-based systems with support for a wide range of network protocols, including Transmission Control Protocol (TCP), Hypertext Transfer Protocol Secure (HTTPS), Domain Name System (DNS), and others (Baykara & Das, 2018).

The connectivity of smart refrigerators to cloud platforms introduces risks such as data leakage, spoofing attacks, and tampering with sensitive information (Vasilomanolakis et al., 2015). Mobile applications designed to interact with these refrigerators can also serve as potential entry points for attackers if their communication protocols, such as TLS (Transport Layer Security), are not implemented securely (Modi et al., 2013). Furthermore, the integration of cameras and other sensors, along with cloud-based data storage, increases the potential for privacy violations if these assets are compromised (Sharma, 2013).

Another critical challenge is securing the firmware that controls the core functions of these refrigerators. As with any IoT device, the firmware represents a high-value target for attackers seeking to exploit vulnerabilities at the system level (Bhuvaneswari & Satheesh, 2015). Compromised firmware could enable attackers to manipulate the device's functionality or install malicious code, posing serious risks to both the user and their network.

Additionally, cloud services play a central role in storing and processing data sent from the refrigerator, which further broadens the attack surface. Attackers can potentially intercept communication between the refrigerator and cloud services, using techniques such as man-in-the-middle (MITM) attacks to gain unauthorized access (Widdowson, 2016).

Penetration testing in this context allows us to simulate potential cyber-attacks and evaluate the system's defenses against real-world threats. In this paper, we will perform a detailed breakdown of the assets and network architecture of the refrigerator system, apply the STRIDE model to assess potential threats, and then conduct penetration tests to highlight any vulnerabilities that may exist.

3. Threat Assessment

An evaluation of the system's vulnerability was necessary prior to moving on.

3.1 Identifying Assets

Passive and active information gathering helped identify assets. The purpose of establishing asset identifiers is to "understand where to focus probable attacks in the interest of time." (Guzman & Gupta, 2017).

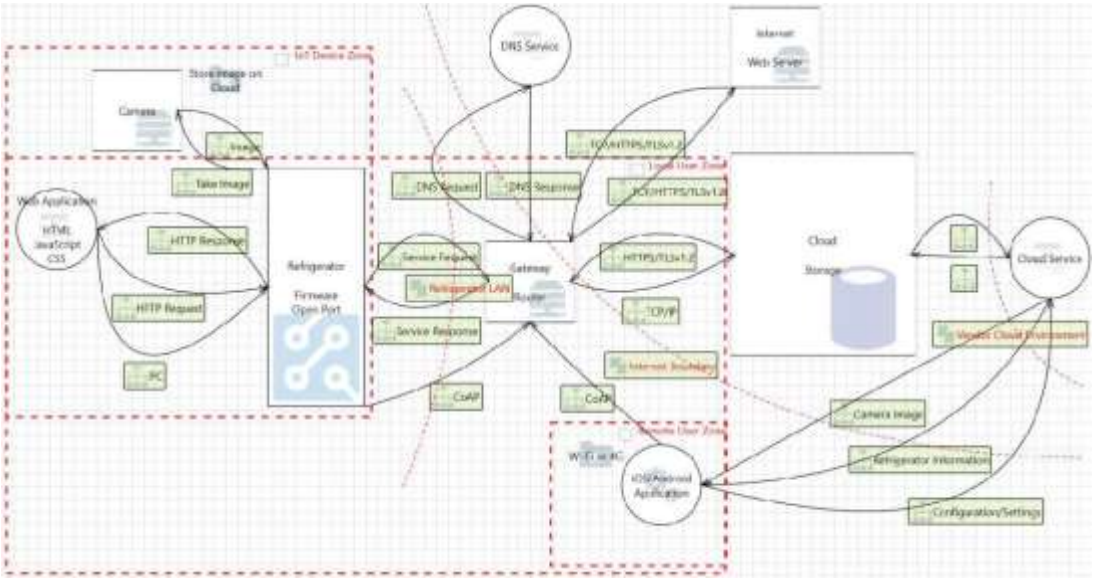
"Table 1: Assets of the system under consideration

ID	Asset	Description
1	Refrigerator	The refrigerator offers a display with the Tizen 5.0 operating system. This system presents the usage of applications – such as an internet browser. Peripherals such as a camera can be located inside the refrigerator which has a feed to an application in the operating system.
		The device and system support a variety of network protocols, such as IP, TCP with TLSv1.2, UDP, DNS, DDNS, MDNS, NTP, ICMP, HTTP, HTTPS and CoAP with DTLSv1.2.
2	Camera	The camera does not provide a live-feed but rather uploads a picture to a cloud-service. This picture is available to applications on the fridge or a secondary device that has been paired with the refrigerator by authentication on the associated platform (Samsung SmartThings).
3	Firmware	he firmware of the refrigerator is foremostly based on the Linux kernel 5.9. System functions and domains are controlled and delegated by the firmware.
4	Web applications	The Tizen operating system offers the usage of the Samsung internet browser. A running webserver on port 17654 has been found but with an unknown service and purpose.
5	Mobile applications	Android and iOS applications are an alternative to controlling various settings in the refrigerator. Also, uploading and downloading media between the mobile and refrigerator. All network traffic is sent and received via the Samsung Cloud API (SmartThings Cloud) through the mobile device's network. The mobile application (Samsung SmartThings) connects to the cloud environment in order to authenticate against the refrigerator. A Samsung account (credentials) is needed to perform any actions and must be paired and matched with the one account used on the refrigerator.
6	Cloud services	The refrigerator uses the Samsung Cloud and SmartThings Cloud to update its current state. It opens a temporary connection (ephemeral port) to the cloud service in order to receive or send a request or reply.
7	Networking	All connections to internet are done through Wi-Fi on the IEEE 802.11 protocol – except for the mobile applications, which could be interconnected to a mobile network.

Now that we know where to find the assets, we can break down the application into its component parts.

3.2 Decomposing the System Under Consideration"

It was necessary to disassemble the system in order to examine the fridge. Finding weak spots or exposed entry points gets easier when you examine the device's dataflow and surroundings.



"Figure 1: The decomposed system of the refrigerator

Following the mapping of the device's dataflow, the attack surface and, more specifically, access points, were documented.

Table 2: The entry points of the refrigerator

ID	Entry point	Description
1	Refrigerator	The refrigerator uses a web browser and compliments a mobile application. The refrigerator uses a cloud service for updates and storage. Open ports of the refrigerator have been discovered and analyzed.
2	Web application	The web application present on the device and operating system uses the TLS 1.2 encryption over the TCP protocol. When performing a HTTP request, the system enforces the usage of TLS 1.2 encryption and is not allowed to downgrade encryption certificates and protocols. The web application supports JavaScript and CSS. Note: The web application is Samsung internet browser
3	Cloud services	The refrigerator sends an image of the physical storage whenever the door has opened and closed. This image is stored on the cloud and could be accessed from the mobile application. This communication craves the usage of a trusted Samsung certificate, presented in Error! Reference source not found.
4	Firmware	The refrigerator runs on a firmware to control the device. This firmware may only be acquired through vendor technical support. Applications can utilize the firmware to manage behavior, only if it is granted by access control.
5	Camera	The camera is operated by the firmware and provides an image when the refrigerator door has closed.
6	Mobile application	The mobile application can monitor the refrigerator and apply configurations. The application must be paired with the refrigerator on the same network with credentials to match with a Samsung account. All the data is fed to the vendor cloud environment to configure and monitor the refrigerator.

7	Networking	All connections to internet are done through Wi-Fi on the IEEE 802.11 protocol – except for the mobile applications, which could be interconnected to a mobile network.”

After the entrance sites were determined, the next step was to determine the risks were associated with them.

3.3 Identifying Threats

We utilized the STRIDE method to identify potential dangers.

“Table 3: Using the STRIDE approach to assess potential threats

ID	Threat type	Description
1	Spoofing	Examine the device and system for threats related to the spoofing of the refrigerator’s identity. Look for ways to potentially overcome trust boundaries. Analyze the authentication and authorization process with the refrigerator’s application interfaces. Evaluate the SmartThings application communication for the capacity to forge requests.
2	Tampering	Assess the refrigerator’s messaging communication between applications and devices. Identify points in the refrigerator that offer an opportunity to tamper with data. Make efforts towards tampering with firmware and applications to perform unauthorized actions.
3	Repudiation	Identify areas where illicit access is allowed without logging. Disable web and mobile application tracing functionalities.
4	Information disclosure	Fuzz application parameters or arguments to impact application error disclosures. Identify open ports with their respective services. Incite confidentiality and integrity in the browser interface. Identify clear text communications. Review usage of HTTP headers and user-agent profile. Pinpoint usages of API endpoints and application backend technologies
5	Denial of service	Identify points where data can be sent to the refrigerator. This could be open ports with no congestion control. Find ways to exhaust or drown out legitimate requests.
6	Elevation of privilege	Examine administrative capabilities of lower application users. Identify weaknesses of segregation in terms of administrative and user-level privileges.”

Keep in mind that the results and assessments from the reconnaissance and discovery stages already allow you to rule out a few paragraphs. One example is the information disclosure heading, which describes a method to "identify clear text communications" by simulating the device's network traffic as it does various activities. This method has been proven and verified.

3.3.1 Relevant Attack Vectors and Documenting Threats

While getting ready to assault the gadget, you'll see a list of potential entry points and ways to exploit it. New information also forms the basis for attack vectors that are relevant. Based on the many parts of the system, the Internet of Things Penetration Testing Cookbook identifies high-level dangers. A potential attacker's high-level threats in this scenario, disregarding any

previous testing that may have been carried out:

- Use vulnerable ports to your advantage.
- Obtain or alter sensitive information by impersonating external services, such as cloud-services or mobile apps.
- Put malicious and bespoke firmware on the fridge.
- To stop legitimate requests from going through, flood the fridge with malicious ones.
- For example, a rogue programme or JavaScript might get access to user and application data.

"Table 4: Abuse open and unsecured ports, Threat #2

Threat description	Access user and application data
Threat target	Samsung internet browser, Refrigerator user.
Attack techniques	An attacker could host an illicit webserver containing malicious HTML and JavaScript to gain credentials – a phishing attack.
Countermeasures	Rightful implementation of the Content Security Framework in the browser.
Scoring (likelihood and impact)	
Cost	Low
Complexity	Low
Reputational impact	Low
Repeatability	Medium
Damage impact	High

Table 5: Spoof external services, Threat #3

Threat description	Access user and application data
Threat target	Cloud-services, Samsung SmartThings.
Attack techniques	Attacker sends and receives requests from a false MAC and IP address that the services trust. An attacker could disguise itself as a cloud-service in order to alter behavior on the device. Attacker analyzes the mobile application to reveal security flaws or information that could offer compromise. An attacker could dismantle the mobile application and modify it in order to uncover vulnerabilities.
Countermeasures	Packet filtering – conflicting IP addresses are blocked, cryptographic network protocols (TLS 1.2, DTLS 1.2),
Scoring (likelihood and impact)	
Cost	Medium
Complexity	Medium
Reputational impact	High
Repeatability	Medium
Damage impact	High"

As mentioned, the next section will grade these functions according to the sensitivity evaluation that takes into account the worst-case scenario threat effect.

3.3.2 Rating Threat Impact and Function Sensitivity

“Scoring the sensitivity of functions" was an issue that NCSC tackled, as covered in Section. The effect and result aspects form the basis of the sensitivity rating system, which has to be addressed. The effects on accessibility, integrity, and secrecy are the primary considerations in this evaluation.

“Table 6: The sensitivity of system functions

Critically sensitive	High and moderate sensitivity
Samsung SmartThings	Samsung internet browser
Cloud-services	Refrigerator network applications/services”

Finally, this episode concludes with an analysis of the accessible attack surface, with a focus on the aims, objectives, and constraints of the thesis.

4. Attack and Gaining Access

Exploitation and payload-based system access is the primary objective of the first penetration test. The theories, methodologies, and approaches used, as well as the assault surfaces and entrance points, are all detailed in this part. The section concludes with the results and discussion of findings -Parts of this episode show the several entry points that the gadget has.

4.1 Access User and Application Data

A locally hosted website will serve as the basis for testing, and the target system enters it. The goal of this webserver is to store malicious scripts and activities that the target machine's web browser (Samsung internet browser) might be susceptible to.

4.1.1 Same-origin Policy Bypass

The need to encapsulate scripts and documents built from a separate origin is a principle that is often seen in browsers. Since the protocols are different, http://www.example.com/ cannot connect with https://www.example.com/. Protocol, hostname, and port origin chain verification is an important part of this policy's proper implementation.

After visiting a malicious website, the victim is tricked into thinking they are opening a legitimate page. However, when they hit a button, a triggered prompt appears, asking for the victim's account and password. Bypassing the same-origin policy, the victim is immediately sent back to the malicious website, where the login and password are displayed, after giving these credentials.

In order to conduct this penetration test, a hosted website must have JavaScript code. The script function has to be set to execute when an HTML or PHP button is clicked.

To begin, the script is executed in a typical manner on the malicious website, without accessing the legitimate website via a tab. This ensures that the JavaScript functionality is tested without attempting to circumvent the same-origin policy. Finally, try opening a new tab

using the window. Open () method instead of the previous one; this time, you're hoping the malicious code is still running in the background.

Regardless of whether redirects were used or not, the execution of the JavaScript failed. It was confirmed that the Content Security Framework and same-origin policy were present since the function seizes to run when a new tab is opened. To ensure an accurate assessment of this policy's existence, certain script adjustments were implemented: linking the malicious website's document. Domain to the real website, in order to encourage the genuine origin to remain constant throughout. However, this had no discernible impact.

The fact that the assault or bypass didn't succeed meant that the browser was current. As mentioned before, this assault relied on a publicly known CVE; nonetheless, it was necessary to verify and authenticate that the browser was really running version 1.0 due to the user-agent fingerprint.

Since an attack might enable hostile websites to utilized scripts in other sites' Document Object Model (DOM), it is crucial to preserve the same-origin policy. The attacker might thus monitor and manipulate data coming from many sources, jeopardising any kind of secrecy.

4.1.2 Open Redirect and Address Bar Spoofing

Unvalidated forwards and redirects are known as open redirects. By combining a redirect with modifications to the document object model (DOM), the attacker may send the victim to a genuine website without validation, but with the validated website still visible in the address bar.

In order to conduct this penetration test, a hosted website must have JavaScript code. A non-interactive PHP or HTML function must be called upon by the script function.

It is possible to run JavaScript when the page has loaded by hooking the HTML website with a <body on page show="redirect Function ()">. By manipulating the document object model (DOM), this script tricks victims into visiting a phished website that seems to be authentic, even down to the address bar.

Without the user's knowledge or consent, the page unexpectedly redirected the internal user after it had loaded. To prevent spoofing, however, the browser removed the address bar whenever the actual website's DOM was modified.

The reroute that was not authorized was successful. Nevertheless, the next steps an enemy may take are unclear. It would just take a convincing redirect to trick the victim into entering credentials and sending them to the attacker. Another thing to keep in mind is that, as an implementational problem, the popup blocker was deactivated by default. As a result, window. Open () might fire automatically, without any input from the user. This problem was resolved after updating or resetting the Samsung web browser. Because this can have been due to flawed software installation or packaging upon delivery, it is difficult to classify it as a vulnerability once it has been patched or addressed.

Despite the fact that this was effective, the idea that it may happen again is unfathomable. Upon searching the internet, no mention of this problem has been made. The confidence in the final conclusion would decrease as a result of using abductive reasoning, leading to a poor success rate.

4.1.3 Further Penetration Testing with BeEF

Using the Browser Exploitation Framework Project (BeEF), the browser was compromised. The hook was able to connect to the fridge via eavesdropping, which goes against the CSP or CSF regulation that says a script source JavaScript tag can't run. There were no discernible outcomes, however, when using various instructions from the BeEF interface.

The camera was attempted to be accessed using Adobe Flash Player and ActiveX. To play media in browsers, you need certain plug-ins, which are developed by third parties. It was not possible to install these extensions because they were not compatible with the device's browser.

Because the hook did not successfully build an invisible frame in the DOM, the browser detected that the attempt to initialize clickjacking had failed.

Although it caused some little UI slowness, the hook was successful in forging DNS queries to a domain resolver.

4.2 Abuse Open and Unsecured Ports

The targeted machine does nothing while it makes various requests to webservers, domain name servers, and devices in its immediate proximity. The following section details the subdomains that have been the target of attempted assaults on their services.

4.2.1 Vulnerability Scan and Brute-force Attack with Nikto

Web servers, like the one identified during the reconnaissance phase (17654/tcp), would respond to HTTP queries with the "404 Not Found" error message. Since this port had not been located yet, Nikto was used to gather a large number of requests for shared services. To do this, we sent queries to the webserver in the expectation of a response unrelated to the "404 Not Found" error.

The brute-force assault included a list of commonly used webserver authentication paradigms and folders.

With 7916 requests, this script may be executed. Wireshark was used to keep tabs on requests and answers while the Nikto script was operating.

```
# nikto -host 192.168.1.134:17654
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.134
+ Target Hostname:    192.168.1.134
+ Target Port:        17654
+ Start Time:         2020-02-25 09:31:01 (GMT-4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the
MIME type
+ No CGI Directories found
+ 7916 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2020-02-25 09:33:23 (GMT-4) (142 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

"Figure 2: Performing a webserver vulnerability scan by brute-force"

There were 7916 queries sent to the webserver, but no responses were received (other than 7916 "404 Not Found" errors), as shown in Figure 10. Even after executing instructions on webservers using CGI directories, the script received no responses. The lack of the X-Frame-Options, X-Content-Type-Options, and X-XSS-Protection headers was, however, detected.

It was premature to draw any conclusions about the long-term effects of an assault that failed to identify this service. Having a webserver operating on an unfiltered open port is unconventional, however. A file structure will inevitably be present in this service. But access to this port's service could still need certain credentials.

Critical headers were found to be missing as a result of the assault. However, since the use is still unknown, it does neither prove or disprove the existence of an exploit in this instance. Visiting web sites does not automatically include these headers. It is very improbable that the absence of headers would be significant given the inactivity of this port and its service when surfing the web.

The directory path, which contains files produced or maintained by the service, would have been exposed by Nikto had this attack been successful. Once inside, the attacker might compromise data by exploring its structure. In principle, it might be brute-forced to success given enough time.

4.2.2 TCP SYN Flood Attack

At all times, the device had TCP ports open and listening, making it vulnerable to denial-of-service or TCP SYN flood assaults.

At the start of a TCP flood assault. Due to the CPU being overloaded and overworked from maintaining connection for each SYN request, valid requests may experience delays or even rejection on the destination system.

The sending computer sends a TCP SYN to start the handshake, and the receiving machine wishes to send a TCP SYN/ACK in response. The receiver will be overwhelmed by the number of requests it needs to handle if it does not have congestion management or pipelining.

While the attacker was hinging the target system, another source attempted to ping the victim on both the internal and external networks in order to measure the delay caused by the denial-of-service attack. Additionally, attempting to see the internet via the web browser of the target system in order to observe any possible effects.

```
# hping3 --syn --rand-source --flood -p 17654 192.168.1.134
```

“Figure 3: hping3 command

The reaction time increased sharply when the floods started:

```
ping 192.168.1.134
PING 192.168.1.134 (192.168.1.134) 56(84) bytes of data.
64 bytes from 192.168.1.134: icmp_seq=33 ttl=64 time=3.62 ms
64 bytes from 192.168.1.134: icmp_seq=34 ttl=64 time=7.55 ms
64 bytes from 192.168.1.134: icmp_seq=35 ttl=64 time=33.4 ms
64 bytes from 192.168.1.134: icmp_seq=36 ttl=64 time=56.4 ms
64 bytes from 192.168.1.134: icmp_seq=37 ttl=64 time=79.4 ms
64 bytes from 192.168.1.134: icmp_seq=38 ttl=64 time=244 ms
...
64 bytes from 192.168.1.134: icmp_seq=39 ttl=64 time=2923 ms
64 bytes from 192.168.1.134: icmp_seq=40 ttl=64 time=3806 ms
64 bytes from 192.168.1.134: icmp_seq=41 ttl=64 time=3826 ms
64 bytes from 192.168.1.134: icmp_seq=42 ttl=64 time=4353 ms
64 bytes from 192.168.1.134: icmp_seq=43 ttl=64 time=13639 ms
64 bytes from 192.168.1.134: icmp_seq=44 ttl=64 time=9804 ms
```

Figure 4: ICMP response times of target device, pinging from internal network

The target device had poor reaction times and timeouts while attempting to utilized network dependent services, but no serious issues, such as a freeze or reboot, were reported

```
ping 130.237.6.49
PING 130.237.6.49 (130.237.6.49) 56(84) bytes of data.
64 bytes from 130.237.6.49: icmp_seq=1 ttl=54 time=1.78 ms
64 bytes from 130.237.6.49: icmp_seq=2 ttl=54 time=1.92 ms
64 bytes from 130.237.6.49: icmp_seq=3 ttl=54 time=2.5 ms
64 bytes from 130.237.6.49: icmp_seq=4 ttl=54 time=3.64 ms
...
64 bytes from 130.237.6.49: icmp_seq=8 ttl=54 time=65.2 ms
64 bytes from 130.237.6.49: icmp_seq=9 ttl=54 time=43.83 ms
64 bytes from 130.237.6.49: icmp_seq=10 ttl=54 time=32.7 ms
64 bytes from 130.237.6.49: icmp_seq=11 ttl=54 time=45.32 ms
```

Figure 5: ICMP response times of WAN, pinging from external network”

There was no seizure of response time from the target machine. Unfortunately, responses to valid requests were delayed, as previously stated. The deluge affected the target computer, not the network or router, as seen in Figure.

There was a little effect on availability from the almost successful TCP SYN flood assault. Unlike most denial-of-service assaults, this one succeeded in keeping the network from being overloaded. On the other hand, the target computer had trouble handling both legitimate requests and every SYN request. While this may not cause alarm, there are ways to reduce congestion, such as by using a firewall or imposing rate limits.

4.2.3 CoAP Resource Discovery Tampering

Devices in the same area are required to share resources by means of multicast under the Constrained application protocol. An adversarial actor might potentially take advantage of the protocol's foundation in a RESTful API, which includes operations like GET, POST, PUT, and DELETE, to modify devices without authorization.

It was necessary to do study on the theory behind the CoAP service and the underlying

technologies that might be utilized to deliver CoAP requests.

It is possible to send fake CoAP messages to the target machine's CoAP service on port 5683/udp using the cf-browser programme from Californium. The target computer has to receive the GET coap://192.168.1.134:5683/.well-known/core message before a resource discovery can be executed.

To get access to a response's resources, one may construct a POST request with a payload that matches the structure of the received resources but changes the values associated with them.

The host computer received a UDP discovery response at the same time as the resource discovery message. Nevertheless, the resources of the intended computer were absent from this reply.

When coaps was present, encrypted communication was enforced by sending the identical message to 5684/udp. However, as the cypher spec protocol remained unchanged, this led to a failed handshake.

Some remarks were made in an effort to alter the request parameters and alternatives. The request may have been structured incorrectly. This is one of several modified requests: sending a PUT request to 192.168.1.134:5683/. well-known/rd?ep=node1 with the following payload: VALUE = </sensors/temp-1>; rt="temperature-c”; if="sensor".

Attempts to modify the requests had the same results as before.

There is some pertinent work on this protocol that shows a basic way to grasp device resources. With this system, it wasn't the case. This thinks about the concept of keying materials and access control lists that the CoAP protocol uses using abductive reasoning. The study on this protocol has come to a close, thus it's safe to assume that there are likely to be many techniques and exploitation routes to access the device's resources. This idea is further supported by the use of CoAPS with DTLS 1.2.

4.3 Spoof External Services

Spoofing is a way to trick and redirect internet traffic to and from a malicious actor who is erroneously believed to be trustworthy, which is useful when the fridge communicates with cloud-services and mobile apps.

4.3.1 Spoofing Against Cloud-services

The attackers set out to verify the legitimacy of the cloud services' permission. By masquerading as a trusted user and updating its MAC address to an existing one, an attacker may get access to the internal network. The next step is to check with the cloud service to verify whether it has the necessary authorization to start transmitting sensitive data and send traffic to a real user (to avoid phone alerts or denial of service). The assault ends by posing as a refrigerator.

The assault was carried out by using the tool Macchanger to alter the MAC address of the adversary's device to the one on the refrigerator. After powering down the fridge and rebooting the network router, the router mistook the fridge for the enemy's equipment, even though it had received the IP address of the fridge. The problem arises, however, when trying to establish a TCP handshake with cloud services.

When trying to verify the identity of a user, some services rely only on the IP address. If the assault had succeeded, the cloud service would have responded to the real IP address and accepted the erroneously obtained one. The legitimate system would be inundated with answers from a multiplicity of services, leading to a distributed denial of service assault, if the attacker were to acquire additional services that operate on the same conditions.

The assault may have also made replay assaults possible, in which the hacker sends out false alerts or otherwise mimics real requests. Unfortunately, this was not feasible since the assault could not authorize the cloud-servers.

4.3.2 Impersonating the Cloud-services

To launch this attack, a man-in-the-middle (MITM) proxy has to be put up between the fridge and the servers in the cloud. The objective of this deed was to divert crucial communications from the refrigerator by making it believe it was communicating with reputable servers while, in reality, an enemy was really doing the surveillance.

In the beginning, the attacker would use the refrigerator as a relay to send all requests to the local Apache server that had a DNS resolver installed. The server had a forged certificate authority that had been signed with the Samsung website because the fridge used SSL/TLS for communication. The Apache server was prepared to accept any cypher suite when the certificates were placed onto it. The results of the SSL/TLS protocol tests were correct, however, since the fridge refused to accept the server's SSL handshake. An attacker could intercept and collect traffic between the cloud-services and the refrigerator if it had accepted the SSL handshake. Then, the adversary might have changed the temperature or uploaded photos to the device.

It is only fair that Samsung has instituted the procedure for validating certificates. Successfully editing man-in-the-middle attacks becomes quite difficult because to this. As a result, a successful assault is considered low-level.

4.3.3 Decompiling and Analyzing the Mobile Application

According to 5.1.2, Samsung SmartThings is an app for mobile devices. The files required to run this software are compressed into the Android Package Kit (APK) format. With the help of apktool and dex2jar, the SmartThings app was decompiled.
To go into the application's source code, one must be proficient in Java and Kotlin, two programming languages that are built on top of the Java Class Library.

There were no obvious security holes discovered during the reverse engineering of the mobile app. Since the application's 6300+ files included several native libraries and archives, it was simple to make a mistake or get confused when inspecting the Java files.

There is a way to debug the app, though: by using APIs in the Android environment and printing variables or helpful information to the user interface. Unfortunately, time constraints and relevancy prevented this from ever being tested.

It would have been catastrophic if an attacker had learned how to use a hardcoded encryption key or another implementation that was carelessly included in the application's source code. An inadequate mobile app may expose users to additional risks like data leakage, API communication, and unsafe data storage. This is very improbable since it did not occur

throughout the investigation.

This chapter presents the threat traceability matrix to enhance readability and understanding, together with the findings of the penetration testing and the threat analysis. With this all-encompassing method, danger actors, impacted assets, assaults, and their surface, purpose, and impact may be effectively encircled.

5. Threat Traceability Matrix

Putting the assaults in context with the recorded incidents given in section.

Reviewing the results and debating their validity, reliability, and applicability in a nutshell. The defense of the system's safety while simultaneously addressing the research issues posed by the goals.

The importance of cyber security is growing. There are moral and financial ramifications to the catastrophic impacts that businesses and other areas might face. The offender may, for instance, access financial accounts or private data or refuse service. The importance of data security is rising in tandem with the expansion of the cyber world. An enormous number of IoTs will validate the need of doing a study that looks at this, as said before.

Research into threat assessment and penetration testing has led to a damning condemnation of the system's security. This research shows that the Content Security Framework and other mitigation strategies reduce vulnerability to typical forms of Cyber-attacks, such as phishing.

Issues with the popup blocker and other minor features were also discussed in this report. Even the most reputable suppliers make errors; this was fixed following an update.

The absence of mitigation measures, such as rate limitations, is evident despite the fact that the DoS assault did not significantly affect availability.

The study that was carried out shown that external services, such as cloud-servers and mobile apps, are secure.

The findings show that both Tizen and Samsung priorities security, particularly privacy. Protecting users' privacy and sensitive information is the primary goal of encryption software and the lawful implementation of security protocols and certifications.

5.1 Sustainability and Ethics

In order to stay on the right side of the law and disclose information responsibly, this thesis has used ethical hacking as its technique. Concerning the moral dimensions, this theory has run smoothly. The attack vectors have been halted by the adoption of mitigation procedures, even if findings may indicate the existence of vulnerabilities. That is to say, no evidence of exploiting these vulnerabilities has been found or shown. where a number of security holes were discovered in 2017, prompting Samsung to substantially beef up Tizen's defenses ever then. In order to stop Cyber-attacks that might harm society, the economy, and the environment, this is the best course of action.

6. Conclusion

The security of smart refrigerator systems, and IoT devices in general, is increasingly critical as they become more integrated into daily life. This paper's comprehensive threat assessment and penetration testing highlight significant vulnerabilities across hardware, software, and cloud components. While the penetration tests revealed some weak points, including open ports and the potential for spoofing and phishing attacks, they also demonstrated that current security measures, such as the STRIDE model and encrypted communication protocols, provide a strong defense against common cyber-attacks. However, there is room for improvement, particularly in the areas of firmware security and protection against denial-of-service attacks. The findings underscore the need for ongoing security updates, robust encryption, and improved firmware protections to safeguard against evolving threats. With proper security practices, smart refrigerators can offer both convenience and safety in a connected world.

References

1. Ali Hedayati (2012), "An analysis of identity theft: Motives, related frauds, techniques and prevention", Journal of Law and Conflict Resolution, Vol. 4, Issue. 1, pp. 1-12.
2. Rajni Misra, Renu Dhir (2012), "Cyber Crime Investigation and Network Forensic System Using Honeypot", International Journal of Latest Trends in Engineering and Technology, pp 34-40.
3. Bhuvaneswari. P, Satheesh Kumar. J (2015), "A Note on Methods Used for Deception Analysis and Influence of Thinking Stimulus in Deception Detection", International Journal of Engineering and Technology, Vol. 7, pp. 109-116.
4. Uma M., Padmavathi Ganapathi (2011), "A survey on various cyber-attacks and their classification", International Journal of Network Security, Vol. 15, No. 5, pp 390-396.
5. Amanda Jane Widdowson (2016), "CHEAT: An updated approach for incorporating human factors in cyber-security assessments", Information & Communications, Security Engineering, pp. 1–7.
6. Vasilomanolakis, Karuppayah. S, M¨uhlh¨auser .M, and Fischer .M (2015), "Taxonomy and survey of collaborative intrusion detection", ACM Computing Surveys, Vol. 47, pp. 1–55.
7. Sang Ni, Mengbo Xie, Quan Qian (2017), "Clustering Based K-anonymity Algorithm for Privacy Preservation", International Journal of Network Security, Vol. 19, No. 6, pp. 1062-1071.
8. Hassen Sallay, Adel Ammar, Majdi Ben Saad and Sami Bourouis (2013), "A real time adaptive intrusion detection alert classifier for high-speed networks", IEEE 12th International Symposium on Network Computing and Applications, pp. 73–80.
9. Muhammet Baykara and Resul Das (2018), "A novel honeypot-based security approach for real-time intrusion detection and prevention systems", Journal of Information Security and Applications, Elsevier, Vol. 41, pp. 103–116.
10. Modi .C, Patel .D, Borisaniya .B, Patel .H, Patel .A, and Rajarajan .M (2013), "A survey of intrusion detection techniques in cloud", Journal of Network and Computing Application, Vol. 36, pp. 42–57.
11. Butun I, Morgera SD, Sankar R (2014), "A survey of intrusion detection systems in wireless sensor networks", IEEE Communications Surveys & Tutorials, Vol. 16, No. 1, pp. 266–282.
12. Chris Moore (2016), "Detecting Ransomware with Honeypot techniques", Cybersecurity and Cyber forensics Conference, pp. 77–81.
13. Shengyi Pan, Thomas Harris (2015), "Developing a Hybrid Intrusion Detection System Using Data Mining for Power System", IEEE, pp. 3104–3113.
14. Abhishek Sharma (2013), "Honeypots in Network Security", International Journal of Technical

Research and Application, Vol. 1, pp. 7–12.
15. Aaditya Jain, Bala Buksh (2015), "Advance Trends in Network Security with Honeypot and Its Comparative Study with Other Techniques", International Journal of Engineering Trends and Technology, Vol. 19, pp. 304–312.
16. Alhakami .W, ALharbi .A, Bourouis .S, Alroobaea .R, and Bouguila .N (2019), "Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection", IEEE Access, Vol. 7, pp. 52181–52190.
17. Lakshmanaprabu .S.K, Shankar .K, Ilayaraja .M, Abdul Wahid Nasir, Vijayakumar Naveen Chilamkurti (2019), "Random forest for big data classification in the internet of things using optimal features", International Journal of Machine Learning and Cybernetics, Vol. 10, pp. 2609–2618.
18. Lakshmanaprabu, S.K., Shankar, K., Gupta, D., Khanna, A., Rodrigues, J.J.P.C., Pinheiro, P.R., & de Albuquerque, V.H.C. (2018), "Ranking analysis for online customer reviews of products using opinion mining with clustering", Complexity, Hindawi, pp. 1–9.
19. Khraisat A, Gondal I, and Vamplew P (2019), "An anomaly intrusion detection system using C5 decision tree classifier", Trends and applications in knowledge discovery and data mining, Springer, pp 149–155.
20. Matthew S.G. (2014), "Predicting crime using Twitter and kernel density estimation", Decision Support Systems, vol. 61, pp. 115–125.