Final Project Report: USB Logging

Matthew Valentino
mattval@terpmail.umd.edu
University of Maryland
College Park, Maryland, USA

Aryan Kakadia
akakadia@terpmail.umd.edu
University of Maryland
College Park, Maryland, USA

Revaant Tandon
rtandon@terpmail.umd.edu
University of Maryland
College Park, Maryland, USA

Joel Joseph
joseph16@terpmail.umd.edu
University of Maryland
College Park, Maryland, USA

ABSTRACT

This project proposes the development of a USB logging application tailored for air-gapped systems, focusing on enhancing security by actively monitoring physical USB ports. The application aims to work in conjunction with intrusion protection systems, providing an additional layer of defence against unauthorized access. The system employs a predefined whitelist to filter out recognized devices, ensuring efficient and relevant notifications. The proposed solution addresses the vulnerabilities of air-gapped systems by incorporating USB logging, anomaly detection, and periodic controlled network connectivity for updating intrusion detection system signatures. The evaluation plan outlines metrics such as detection rate, response time, and data exfiltration detection to assess the effectiveness of the USB logging product.

KEYWORDS

USB, Final Project, Logging

ACM Reference Format:
Matthew Valentino, Aryan Kakadia, Revaant Tandon, and Joel Joseph. 2023. Final Project Report: USB Logging. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the realm of cybersecurity, air-gapped systems serve as a critical defence mechanism against external threats. However, recent sophisticated attacks have exploited vulnerabilities in these systems, emphasizing the need for innovative solutions. This project introduces a USB logging application designed to monitor and detect unauthorized access on air-gapped networks. The system aims to enhance security by actively monitoring physical USB ports and alerting administrators to potential threats. The proposed solution integrates seamlessly with existing intrusion protection systems and introduces a whitelist to minimize unnecessary notifications for authorized devices. The motivation behind this project stems from the recognition that while USB logging is not a new concept, its evolution in recent years has been limited. The project addresses this gap by incorporating anomaly detection, whitelisting, and periodic network connectivity for system updates.

2 RELATED WORK

2.1 Literature Review

The increasing reliance on USB devices in our digital world has unfortunately made them a target for various cyberattacks. This project was inspired after reviewing several key research articles on USB-based attacks and exploring their techniques, impacts, and potential countermeasures.

2.2 vulnerability Landscape

Several papers focused on the current state of security weaknesses and exploitable flaws in various systems, software, and technologies. They examined potential threats and attack vectors that attackers can leverage to compromise systems and steal data. These papers detailed different types of vulnerabilities, their prevalence and severity, as well as the security measures in place to combat them

[9] provide a comprehensive overview of USB-based attacks, categorizing them into four groups: storage-based, human-factor, network-based, and firmware-based. They highlight the diverse range of attack vectors and potential consequences, including data exfiltration, malware installation, and system disruption.

[4] propose USB-Watch, a hardware-assisted framework for detecting insider threats through USB activity monitoring. This emphasizes the need for proactive monitoring and detection mechanisms to address insider threats.

[1] discuss USB artifact analysis using Windows event logs, registry, and file system logs. This highlights the importance of forensic analysis in identifying and investigating USB-based attacks. This paper was one of our main starting points when considering the viability of our product.

2.3 Emerging Threats and Detection Methods

A few papers were concerned with newer or more recently discovered vulnerabilities, that haven't been widely observed or addressed by traditional security solutions, making them particularly dangerous. They discussed these methods and the new and innovative methods needed to effectively identify and respond to them.

[6] explore side-channel threats associated with USB-powered devices, focusing on electromagnetic emanations that can leak sensitive information. This study emphasizes the need for novel detection methods that go beyond traditional software-based approaches. They also present a method for detecting USB storage device behaviors using electromagnetic emanations. This demonstrates the potential of non-invasive techniques for identifying suspicious USB activity and influenced how our product approached detection.

[2] analyze Windows artifacts generated by USB storage devices, emphasizing the importance of understanding device logs and file system changes to identify potential attacks.

2.4 Addressing Air-Gapped Networks

These papers detailed the system aspect of our product and explored air-gapped systems which are often isolated from external networks and traditionally considered the ironclad fortresses of cybersecurity. They examined the unique vulnerabilities of these systems and the specific strategies needed to protect them from cyberattacks.

[7] analyze hacking techniques for air-gapped computers and propose potential solutions. This research highlights the vulnerability of even isolated systems and emphasizes the need for multi-layered security approaches.

[3] discuss secure USB practices in their chapter on domain security operations. This provides practical guidance for organizations on mitigating USB-based threats.

[5] introduces "POWER-SUPPLaY," a novel attack that leaks sensitive data from air-gapped systems by exploiting power supplies. This research demonstrates the ingenuity of attackers and underscores the need for continuous vulnerability assessments and security updates.

[8] review various attack methods on air-gapped systems, emphasizing the need for comprehensive security measures beyond physical isolation.

These papers proved that USB-based attacks present a significant and evolving threat, requiring multifaceted approaches for mitigation. They highlighted the importance of understanding the diverse landscape of USB-based attacks and their potential impacts and implementing proactive detection and monitoring mechanisms, including both software and hardware-based solutions. Organizations and individuals can better protect themselves from the dangers of USB-based attacks and ensure the security of their sensitive information by utilizing forensic analysis to investigate and attribute USB-related incidents, adopting secure USB practices and educating users about potential threats. Continuous assessment of and updates to security measures will help address emerging threats and vulnerabilities, even in air-gapped environments.

2.5 Limitations of Existing Work

While current security practices for air-gapped systems incorporate various measures, several limitations persist. The lack of external connectivity in air-gapped systems restricts the effectiveness of traditional IDSs, and resource constraints hinder the implementation of resource-intensive intrusion detection systems. The absence of regular communication with central servers impedes the timely update of threat intelligence and detection rules, diminishing the overall security posture of air-gapped systems. Additionally, the current solutions lack a comprehensive focus on USB-related security, providing an opportunity for the proposed USB logging application to fill this gap.

This project aims to overcome these limitations by introducing a USB logging application that specifically targets physical intrusions through USB devices. The integration of anomaly detection, whitelisting, and controlled network connectivity addresses the challenges posed by the unique characteristics of air-gapped systems, providing a more robust and tailored solution to enhance security in critical environments.

3 PROPOSED METHOD/DESIGN

This section outlines the challneges and considerations encountered while devising a USB detection solution. It delves into issues such as diverse USB device types, driver dependencies, and more. It explains each part of our design and how they all function together to produce a cohesive product that would benefit any user of it.

3.1 Challenges

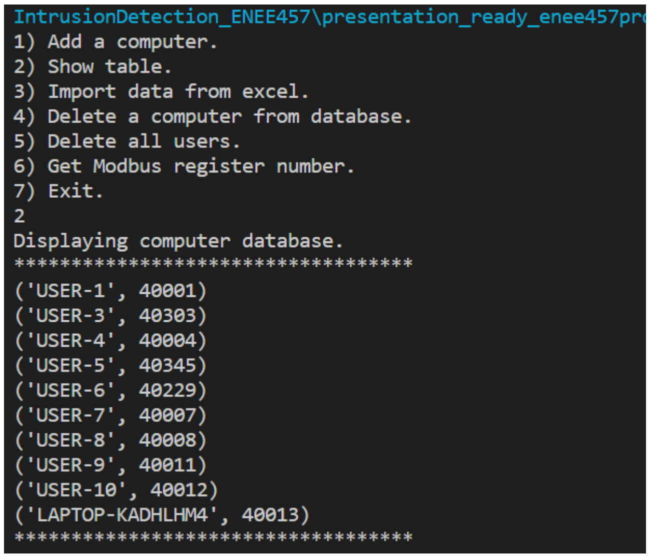
Some of the challenges we wanted to address with our solution were device diversity, driver dependencies, timely detection, high traffic, data protection, and operating system updates. When dealing with USBs, there are so many different types that you may encounter when dealing with the issue of USB detection, there are keyboards and mice, storage devices, and even microcontrollers that you may have to worry about. So one of the big discussions we had was about handling these different devices, and whether some should be blocked altogether or not. Next, we had the issue of driver dependencies, since some devices needed drivers to function correctly, we brought that up as a potential issue, however with the way we proposed handling the connections this proved to not be an issue. Our primary challenge in developing a solution stemmed from achieving swift detection. In any corporate setting, swift and immediate identification of any device connection within the network is crucial. Adelay of even 30 seconds opens a window wherein an individual could potentially install a malicious package onto a company's server. This demand for near-instantaneous detection was our paramount conver throughout the solution's design phase. Another problem we discussed but were not able to test is validity was the scalability of this product and how it handles large scale networks. We were not able to test this since we did not have a large scale network to install this software onto, we were unable to determine the feasibility of this product on a large scale network. Finally, the method we used to implement our detection software should be unaffected by operating system updates. The only change that may affect our product is if in the update they change the device codes we would just have to update those in our code to reflect the changes.

3.2 Our Design

Our proposed solution is built with the user in mind. We wanted to provide an experience that would be as useful to the end user as possible. In doing this we wanted to create a full user interface to give a visual representation of the system and any issues that may arise. During setup, the system administrator would be responsible for setting up the Whitelist/Blacklist. In setting up the whitelist, you would have a list of the device IDs that correspond to the USB devices, when one of these devices is connected to a computer on the network there is no log reported to the system administrator. When setting up the blacklist, we do a similar setup where we put the device IDs in a list, and if the device is connected we want to send an urgent message to the system administrator and block the device from doing anything further once detected. With every other device ID we find, we report a connection to that computer to the system administrator.

Our proposed method of collecting these USB events is by reading through the system logs and finding logs that match a certain code. This code is a system-defined code that allows you to use regular expressions to parse the system log and find the messages you want. We would then take those messages and check the identifier of the USB and compare it to our user defined lists.

Some of the things we may want to consider in the future include an emergency power supply to the administrator's computer to ensure the system is always under watch and to not have the system unmonitored at any time. Additionally, we would like to add security measures to ensure that the communication protocols are secure and encrypted. This would ensure that if an adversary was able to access the system, they would not be able to view the messages and determine a weakness. Finally, we want to ensure that our software is compatible with all Windows versions in operation wherever this product is employed.



```
IntrusionDetection_ENEE457\presentation_ready_enee457pr
1) Add a computer.
2) Show table.
3) Import data from excel.
4) Delete a computer from database.
5) Delete all users.
6) Get Modbus register number.
7) Exit.
2
Displaying computer database.
**********************************
('USER-1', 40001)
('USER-3', 40303)
('USER-4', 40004)
('USER-5', 40345)
('USER-6', 40229)
('USER-7', 40007)
('USER-8', 40008)
('USER-9', 40011)
('USER-10', 40012)
('LAPTOP-KADHLHM4', 40013)
**********************************
```

Figure 1: A snippet of the startup screen of our product

3.3 Challenges With the Current Design

One of the challenges that we faced when creating this product included the ability to quickly detect these USB connection events. When we started our code, it would take up to 30 seconds to get the logs, read them, and then process the data for a possibly malicious piece of hardware. In the end, we were able to refine and enhance our code to bring our time down from 30 seconds to a couple of seconds. While this is not perfect, it is a major improvement and demonstrates the feasibility of the product as well as the utility of it.

Additionally, we faced the issue of device lists. In attempting to implement the White/Blacklist, we wanted to just include a list of the specific device IDs that correspond to the USBs that we want to allow. However, when attempting to implement the lists we ran into the issue of not being able to distinguish between different USB devices due to the shared protocols among these devices. . This warranted an ongoing investigation into ways that we may be able to create these lists and give them the correct functionality.

Finally, we faced the issue of creating and connecting a SQL database to the code. We are currently working on adding that functionality to our product, and are hoping to see this working in its entirety before the end of the semester. The difficulty was with getting everything to communicate properly, and figuring out how to set up a SQL database to store the logs and all of the data we wanted. We are working to figure out what parts of the data are necessary to store and what data may be ignore. Once we get this added, our product would be more scalable for large businesses since they could also compare new alerts to previous alerts to determine any recurring issues/employees.

4 IMPLEMENTATIONS

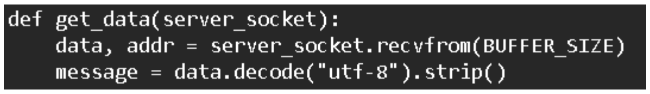
This section provides an exhaustive breakdown of the utilized languages, encompassing libraries, versions,frameworks, and tools. It includes comprehensive explanations of critical code segments, insights into computational techniques, and outlines our testing methodologies. Each component will be individually addressed in separate sections, offering meticulous descriptions and thorough explanations.

4.1 Technology Stack

For this project, we used Python and SQL as our chief coding languages, and Windows as our operating system. In Python, we used the libraries socket, re, and sqlite3. We used socket for communications with WinCollect and receiving system logs. This enabled us to read through the system logs. Additionally, the library re, which is short for regular expressions, which enabled us to use regular expressions for parsing all of the system logs for information on USB connections. Finally, we imported the package sqlite which allows us to access the SQL database and modify/read the database entries. This is important in storing the data to use or viewing past data to compare incidents to. For the SQL segment, we leveraged hte power of SQL to create a comprehensive database housing event logs, meticulously organized into distinct sections and sorted through various methods. SQL functionality facilitated the identification of USB connections, enabling us to conduct comparative analysis against other connections. This capability empowered us to discern and scrutinize USB connections within the dataset, offering valuable insights for our analysis.

4.2 Code Implementations

In this section we will discuss the critical parts of our code and what they do. One of the most important parts of our code is the get data function as shown below.

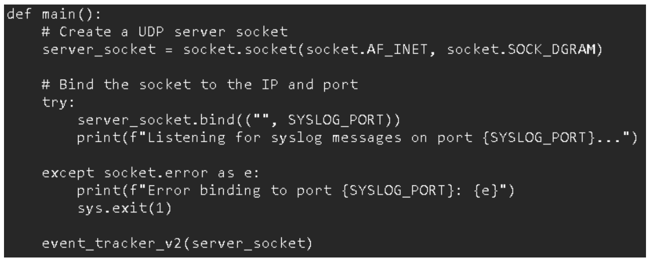


```
def get_data(server_socket):
    data, addr = server_socket.recvfrom(BUFFER_SIZE)
    message = data.decode("utf-8").strip()
```

Figure 2: A snippet of the startup screen of our product

In this segment, our approach involves utilizing the socket library to retrieve data sourced from the Windows log buffer. Once obtained, the data undergoes a decoding process into UTF-8 encoding, enabling its transformation into a human-readable format. Subsequently, we meticulously refine and streamline the content, extracting the essential components while stripping away extraneous elements. This curated information is then prepared and returned, poised for further parsing and analysis within our system.

The other critical part to our code comes in our main function as shown below.



```
def main():
    # Create a UDP server socket
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bind the socket to the IP and port
    try:
        server_socket.bind(("", SYSLOG_PORT))
        print(f"Listening for syslog messages on port {SYSLOG_PORT}...")

    except socket.error as e:
        print(f"Error binding to port {SYSLOG_PORT}: {e}")
        sys.exit(1)

    event_tracker_v2(server_socket)
```

Figure 3: A snippet of the startup screen of our product

Within this function, our primary objective is to set up the UDP server socket, subsequently binding it to the designated IP and USB port. This configuration allows us to actively listen for incoming USB connections on that specific port. Upon detecting a USB connection, we redirect the flow to our event tracker.

Within the event tracker, our focus lies in parsing the assorted system logs to extract specifically the USB-related logs. Subsequently, these logs undergo scrutiny against our predefined white and blacklist parameters. If the identified connection doesn't align with either list or appears on the blacklist, we trigger two actions simultaneously: reporting the event to the system administrator and generating a corresponding entry within our SQL database.

This systematic process ensures a real-time assessment of USB connections, enabling us to swiftly flag and address any unauthorized or potentially harmful activities, thereby bolstering our system's security measures.

4.3 Testing

Our testing methodology primarily involved practical user-based assessments. Operating the program, we systematically connected every USB device available to our computers, capturing the time of detection. On average, our system exhibited a response time of 6-7 seconds post-insertion for device detection. This hands-on approach proved to be the most straightforward and effective way to conduct testing since generating tests directly interfacing with system logs manually proved unfeasible.

Looking ahead, our aspirations include validating our product's functionality in environments like air-gapped systems and expansive network settings, areas we couldn't access for testing during our initial phase. Moreover, we aim to implement automated checks to verify the integrity of our SQL database, ensuring accurate device logging and proper timestamp records.

While our testing was comprehensive within a confined environment, scaling up remains contingent on conducting trials within larger, more diverse networks. This strategic approach will enable us to refine and optimize our product's performance before wider implementation.

5 EVALUATION

The evaluation of our product encountered certain constraints within the confines of the provided examples. Since our focus didn't involve handling passwords or implementing machine learning algorithms, we lacked a specific dataset for evaluation purposes. Furthermore, comparing our product with existing alternatives proved challenging due to the predominant presence of paid software in the market. However, our product effectively operates on devices using the Windows operating system, and we have intentions to broaden its compatibility to encompass Mac and Linux systems, thereby expanding its reach.

Despite these limitations, our evaluation encompassed diverse USB types, each of which interacted seamlessly with our software. We extensively tested storage devices, peripherals like mice and keyboards, and various combinations thereof, affirming our software's versatility.

In our evaluation process, we gauged the detection rate, averaging between 6 to 7 seconds, alongside the immediate response time upon detection. Additionally, a key focus was placed on usability, emphasizing a design approach aimed at readability and accessibility. Our intent was to craft a user interface that caters not only to computer science experts but also to individuals with varying degrees of technical knowledge, ensuring a user-friendly experience for all.

6 CONCLUSION

This project successfully explored the development of a USB logging application tailored for air-gapped systems, addressing the critical need for enhanced security against unauthorized physical access. The proposed solution leverages active monitoring of USB ports, integrates seamlessly with existing intrusion detection systems, and utilizes whitelisting to minimize unnecessary notifications. By addressing the limitations of existing solutions, such as the lack of regular connectivity and resource constraints, this application provides a robust and comprehensive approach to USB-related security in air-gapped environments.

7 FUTURE WORK

While the project has achieved significant progress, several avenues remain for future exploration:

- Scalability and performance: Testing the application on larger networks and optimizing its performance for handling high traffic volumes are crucial steps for real-world deployment.
- Extended detection capabilities: Integrating anomaly detection algorithms and advanced forensic analysis techniques can further enhance the system's ability to identify and respond to suspicious activity.
- Improved whitelist and blacklist management: Implementing user-friendly interfaces and automated device recognition techniques can simplify the creation and maintenance of whitelists and blacklists.
- Countermeasures and mitigation strategies: Exploring proactive measures beyond logging and notification, such as device lockdown or data encryption, can add another layer of defence against USB-based attacks.
- Integration with intrusion detection systems: Investigating deeper integration with existing IDS platforms can facilitate comprehensive threat intelligence sharing and incident response coordination.
- Exploring alternative data sources: Examining additional data sources beyond system logs, such as network traffic analysis or electromagnetic emanations, can potentially reveal hidden attack vectors.
- Continuous vulnerability assessment: Maintaining a research and development focus on emerging USB-based threats and vulnerabilities is crucial for ensuring the application's long-term effectiveness.

By pursuing these future works, this USB logging application can evolve into a powerful and adaptable tool for safeguarding air-gapped systems from the ever-evolving landscape of cyber threats.

REFERENCES

[1] Neyaz A and Shashidhar N. 2019. USB Artifact Analysis Using Windows Event Viewer, Registry and File System Logs. *Electronics* 8, 11 (2019), 1322. https://doi.org/10.3390/electronics8111322

[2] Harlan Carvey and Cory Altheide. 2005. Tracking USB storage: Analysis of windows artifacts generated by USB storage devices. *Digital Investigation* 2, 2 (2005), 94–100. https://doi.org/10.1016/j.diin.2005.04.006

[3] Eric Conrad, Seth Misenar, and Joshua Feldman. 2017. Chapter 7 - Domain 7: Security operations. In *Eleventh Hour CISSP® (Third Edition)*. Syngress, 145–183. https://doi.org/10.1016/B978-0-12-811248-9.00007-3

[4] K. Denney, L. Babun, and A.S. Uluagac. 2020. USB-Watch: a Generalized Hardware-Assisted Insider Threat Detection Framework. *Journal of Hardware Systems and Security* 4 (2020), 136–149. https://doi.org/10.1007/s41635-020-00092-z

[5] M. Guri. 2023. POWER-SUPPLaY: Leaking Sensitive Data From Air-Gapped, Audio-Gapped Systems by Turning the Power Supplies into Speakers. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 313–330. https://doi.org/10.1109/TDSC.2021.3133406

[6] Hao Liu, Riccardo Spolaor, Federico Turrin, Riccardo Bonafede, and Mauro Conti. 2021. USB powered devices: A survey of side-channel threats and countermeasures. *High-Confidence Computing* 1, 1 (2021), 100007. https://doi.org/10.1016/j.hcc.2021.100007

[7] Raja Muthalagu and Vrinda Sati. 2023. Analysis on Hacking the Secured Air-Gapped Computer and Possible Solution. *Cybernetics and Information Technologies* 23 (2023), 124–136. https://doi.org/10.2478/cait-2023-0017

[8] M. T. Naz and A. M. Zeki. 2020. A Review of Various Attack Methods on Air-Gapped Systems. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. 1–6. https://doi.org/10.1109/3ICT51146.2020.9311995

[9] Nir Nissim, Ran Yahalom, and Yuval Elovici. 2007. USB-based attacks. *Computers and Security* 50 (2007), 675–688. https://doi.org/10.1016/j.cose.2017.08.002