A project report on

PEER TO PEER CONNECTION FIREWALL

ABSTRACT

The Python-based program we've developed is a powerful network monitoring and security analysis tool with an advanced file analysis component. It seamlessly integrates with Wireshark to capture real-time data from peer-to-peer (P2P) connections. Users have fine-grained control over data capture, allowing them to filter and collect specific P2P protocols or data streams.

Once data is captured, the program efficiently exports it to a designated SMB (Server Message Block) share, ensuring centralized storage for easy access and management. The dataremains secure and intact during the export process.

An exceptional feature of this program is its integrated virus analysis module, which utilizesthe VirusTotal API v2 for file analysis. This module examines exported data files and checks them against the extensive VirusTotal database, which contains a wealth of information aboutknown malware signatures. By leveraging this API, the program significantly enhances its ability to identify and flag potential security threats, whether they are established malware or emerging risks.

In the event that the analysis module detects a potential threat, the program promptly generates detailed alerts, offering insights into the nature of the threat, its source, and the potential impact on the network. These alerts empower network administrators and security professionals to take immediate action to mitigate risks and protect their infrastructure.

To facilitate historical reference, auditing, and forensic analysis, the program maintains comprehensive logs that document all network activities, data captures, exports, analysis results, and alerts.

Moreover, users have the flexibility to configure the program to align with their specific network and security needs. This includes setting custom capture filters, defining SMB export locations, and adjusting the sensitivity levels of virus detection to match their desired security posture.

Keywords— Network monitoring, Security analysis tool, Wireshark integration, P2P connections, Data capture, SMB (Server Message Block), VirusTotal API, Malware detection, Alerts

I. INTRODUCTION

In an era where digital communication networks are at the heart of our daily lives, ensuring the security and integrity of these networks is of paramount importance. To addressthis critical need, we present an innovative Python-based program, designed to be a formidable tool for advanced distributed peer-to-peer (P2P) communication network security,traffic monitoring, and connection control. This project offers a comprehensive suite of features and capabilities that empower network administrators and security professionals to proactively protect their infrastructure while maintaining the flexibility to adapt to specific network and security requirements.

At its core, this program seamlessly integrates with Wireshark to capture real-time data from P2P connections, providing insights into network activities and traffic patterns. What sets it apart is its fine-grained data capture control, allowing users to selectively filter and collect specific P2P protocols and data streams, making it adaptable to the diverse needs of network administrators.

One of the standout features of this program is its ability to export captured data securely to a designated Server Message Block (SMB) share. This centralized storage ensures easy access and streamlines data management, essential for analyzing historical network activities and incidents.

To bolster network security, the program incorporates an integrated virus analysis module that harnesses the power of the VirusTotal API. This module scans exported data files, cross-referencing them with an extensive database of known malware signatures. By doing so, it significantly enhances the program's ability to identify and flag potential security threats, whether they are established malware or emerging risks.

In the event of a security breach or potential threat detection, the program responds with precision, promptly generating detailed alerts. These alerts offer insights into the nature of the threat, its source, and the potential impact on the network, enabling rapid response and mitigation.

In addition to real-time monitoring and alerting, the program keeps a meticulous record of all network activities, data captures, exports, analysis results, and alerts. These comprehensive logs are invaluable for historical reference, auditing, and forensic analysis, ensuring that network administrators have a complete view of their network's security and performance history.

Moreover, the flexibility of the program is a distinguishing factor. Users can fine-tune the tool to meet their specific network and security needs. This includes setting custom capture filters, defining SMB export locations, and adjusting the sensitivity levels of virus detection, enabling them to align the program with their desired security posture.

By combining Wireshark integration with these advanced features, this Python program empowers network administrators to gain deep visibility into P2P network activity, proactively manage traffic, and effectively safeguard their networks from emerging threats. It's a comprehensive solution for any organization seeking to maintain a secure and well-controlled P2P network environment.

PROPOSED FRAMEWORK

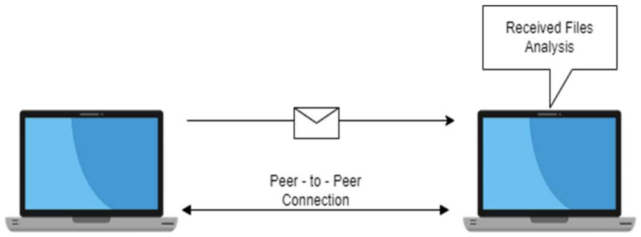
\

A. Data Capture Module:

At the heart of the system is a Python script responsible for capturing data from peer-to-peer (P2P) connections established over a Wi-Fi network. The script utilizes the powerful capabilities of Wireshark to capture real-time network traffic, including the exchange of data, packets, and communications. This comprehensive data capture module not only records the raw data but also offers fine-grained control, allowing users to selectively filter and collect specific P2P protocols or data streams. This level of control is invaluable for targeted analysis and monitoring of network activities.

B. Data Storage and Export:

Once the data is captured, the system stores it securely in the widely supported pcapng (Packet Capture Next Generation) file format. This storage format ensures data integrity and allows for the retention of captured data in an organized and accessible manner. To further streamline data management, the system provides a mechanism for exporting the captured data to a designated Server Message Block (SMB) share. This centralized storage simplifies access and management, providing a repository for historical network traffic data and analysis.



C. File Extraction from SMB Share:

To facilitate the retrieval of specific files from the SMB share, the system incorporates a dedicated Python script for file extraction. This script efficiently extracts files from the shared storage while maintaining data integrity. This extraction process ensures that files are preserved and ready for analysis.

D. Virus Analysis Module:

One of the standout features of this system is its integrated virus analysis module, which employs the VirusTotal API to enhance the security analysis of files. When a file is extracted from the shared folder, it is subjected to thorough scrutiny using this module. The VirusTotal API accesses an extensive database of known malware signatures and checks the file for potential security threats. This real-time analysis significantly augments the system's ability to identify and flag potential risks, whether they are established malware or emerging threats.

E. Threat Response:

If the analysis module detects a potential threat within a file, the system reacts swiftly and decisively. It generates detailed alerts that provide insights into the nature of the threat, its source, and the potential impact on the network. These alerts empower network administrators and security professionals to take immediate action to mitigate risks and protect their infrastructure. Depending on the severity of the threat, the system may offer multiple response options, including isolating or quarantining the infected file, notifying relevant personnel, and taking actions to prevent the threat from spreading further.
In cases where a file is confirmed to be infected, the system has the capability to automatically delete the file from the shared folder, preventing it from posing a security risk within the network.

F. Logging and Documentation:

For historical reference, auditing, and forensic analysis, the system maintains comprehensive logs. These logs document all network activities, data captures, exports, analysis results, alerts, and responses. These records are invaluable for post-incident analysis, compliance reporting, and ensuring transparency and accountability in network security management.

G. User Configuration and Customization:

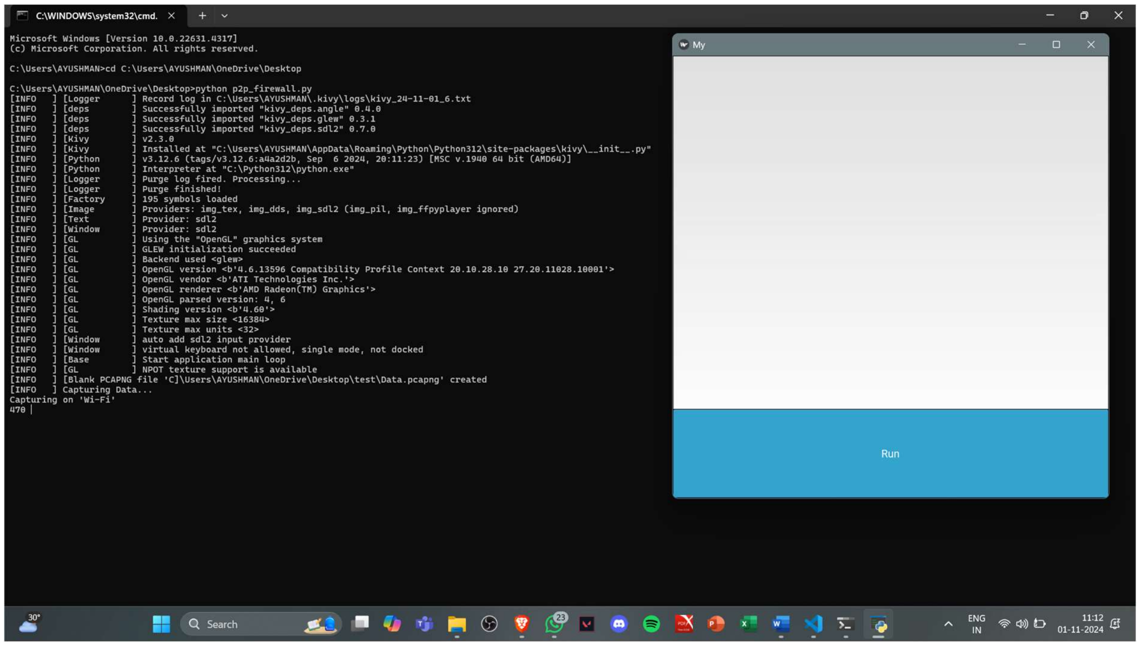
To cater to the specific needs and security requirements of users and organizations, the system offers a high degree of flexibility and configurability. Users can customize the system by setting custom capture filters to specify the types of data they want to monitor. Additionally, they can define SMB export locations and adjust the sensitivity levels of virus detection to match their desired security posture, making the system adaptable to a wide range of network environments and security policies.
In summary, this system provides a comprehensive solution for advanced P2P data capture, security analysis, and network protection. It ensures the seamless monitoring and analysis of

network activities, with an integrated virus analysis module that enhances threat detection. The system's robust threat response mechanism and flexibility make it a powerful tool for safeguarding network infrastructure and responding effectively to potential security threats.

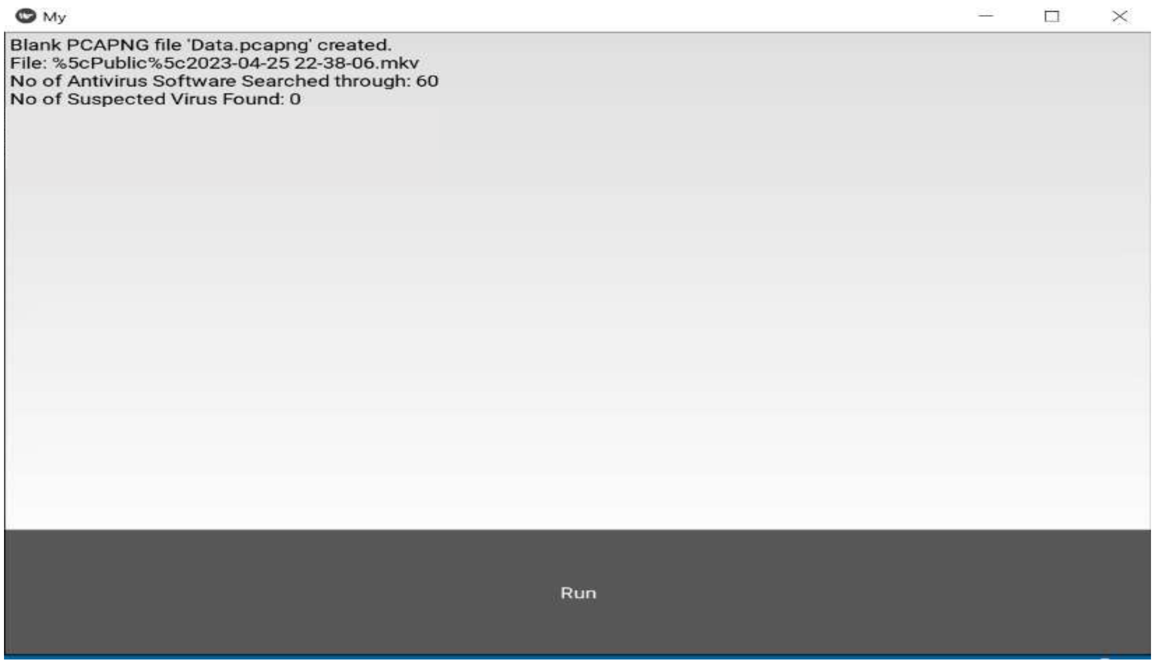
EXPERIMENTAL RESULTS

```
import requests, json, os, subprocess, time, sys
from colorama import Fore, Style
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.textinput import TextInput
from kivy.uix.boxlayout import BoxLayout
from kivy.clock import Clock
from kivy.logger import Logger


API_KEY = "YOUR_API_KEY_HERE"  # Replace with your VirusTotal API key
NETWORK_INTERFACE = "YOUR_NETWORK_INTERFACE"  # Replace with your network
interface (e.g., "Wi-Fi", "Ethernet")
PROTOCOL = "YOUR_PROTOCOL"  # Replace with the protocol used (e.g., "smb")
DIRECTORY_PATH = r"YOUR_DIRECTORY_PATH"  # Replace with the directory path
(e.g., r"C:\path\to\test")


# Function to get file report data from VirusTotal
def getFileReportData(filename, apikey):
    url = 'https://www.virustotal.com/vtapi/v2/file/scan'
    params = {'apikey': apikey}
    files = {'file': (filename, open(filename, 'rb'))}
    response = requests.post(url, files=files, params=params)
    return response.json()

# Function to get file report from VirusTotal
def getFileReport(resource, apikey):
    url = 'https://www.virustotal.com/vtapi/v2/file/report'
    params = {'apikey': apikey, 'resource': resource}
    responseReport = requests.get(url, params=params)
    if responseReport.status_code == 204:
        print("Rate limit exceeded, sleeping for 15 seconds")
        time.sleep(15)
        return getFileReport(resource, apikey)
    return responseReport.json()

# Get list of files to analyze in the specified directory
def getFiles(directoryPath):
    files = os.listdir(directoryPath)
    return [file for file in files if os.path.isfile(os.path.join(directoryPath,
file)) and not file.endswith(('.ini', '.File', ''))]

# Capture data and save to a PCAP file
def captureFileAndData(outputFile, interface):
    print("Capturing Data...")
    Logger.info("Capturing Data...")
    tshark_path = r'C:\Program Files\Wireshark\tshark.exe'
    command = [tshark_path, '-i', interface, '-w', outputFile]
    try:
        subprocess.run(command)
    except KeyboardInterrupt:
        print("\nStopped by User. Goodbye!")
        Logger.info("\nStopped by User. Goodbye!")

# Extract objects from PCAP using Tshark
def extractObjects(protocol, inputFile):
    output_directory = f'{protocol}Objects/'
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)
    tshark_path = r'C:\Program Files\Wireshark\tshark.exe'
    command = [
        tshark_path,
        '-r', inputFile,
        '--export-objects', f'{protocol},{output_directory}'
    ]
    subprocess.run(command)

# Main process to scan files and log/report results
def main(apikey, protocol, interface, directory_path):
    pcapngFile = os.path.join(directory_path, "Data.pcapng")  # Store PCAP file
in specified directory
    directory = f'{protocol}Objects'
    if not os.path.exists(directory):
        os.makedirs(directory)
    with open(pcapngFile, "wb") as pcap_file:
        pass
    print(f"Blank PCAPNG file '{pcapngFile}' created.")
    Logger.info(f"Blank PCAPNG file '{pcapngFile}' created")
    captureFileAndData(pcapngFile, interface)
    extractObjects(protocol, pcapngFile)

    for file in getFiles(f'{protocol}Objects'):
        try:
            file_path = os.path.join(directory_path, file)
            report = getFileReport(getFileReportData(file_path,
apikey)['resource'], apikey)
            total = report["total"]
            positives = report["positives"]
            if positives > 0:
                print(f"File: {file}")
                print(f"No of Antivirus Software Searched through: {total}")
                print(f"No of Suspected Virus Found: {positives}")
                print(f"Deleting file {file} from {file_path}")
                os.remove(file_path)
                Logger.info(f"File: {file}")
                Logger.info(f"No of Antivirus Software Searched through:
{total}")
                Logger.info(f"No of Suspected Viruses Found: {positives}")
                Logger.info(f"Deleting file {file} from {file_path}")
            else:
                print(f"File: {file}")
                print(f"No of Antivirus Software Searched through: {total}")
                print(f"No of Suspected Virus Found: {positives}")
                Logger.info(f"File: {file}")
                Logger.info(f"No of Antivirus Software Searched through:
{total}")
                Logger.info(f"No of Suspected virus Found: {positives}")
        except Exception as e:
            print(f"Error: {e}")
            Logger.info(f"Error: {e}")

    subprocess.run(['rmdir', '/s', '/q', f'{protocol}Objects'], shell=True)
    os.remove(pcapngFile)

# Kivy App to run the analysis and display output
class MyApp(App):
    def build(self):
        layout = BoxLayout(orientation='vertical')
        self.output = TextInput(readonly=True, size_hint=(1, 0.8))
        button = Button(text='Run', size_hint=(1, 0.2))
        button.bind(on_press=self.run_main)
        layout.add_widget(self.output)
        layout.add_widget(button)
        return layout

    def run_main(self, instance):
        sys.stdout = self
        Clock.schedule_once(lambda dt: main(
            API_KEY,
            PROTOCOL,
            NETWORK_INTERFACE,
            DIRECTORY_PATH
        ), 0)

    def write(self, text):
        self.output.text += text
        self.output.cursor = (0, len(self.output._lines))

    def flush(self):
        pass

if __name__ == '__main__':
    MyApp().run()
```

OUTPUT



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AYUSHMAN>cd C:\Users\AYUSHMAN\OneDrive\Desktop

C:\Users\AYUSHMAN\OneDrive\Desktop>python p2p_firewall.py
[INFO   ] [Logger      ] Record log in C:\Users\AYUSHMAN\.kivy\logs\kivy_24-11-01_6.txt
[INFO   ] [deps        ] Successfully imported "kivy_deps.angle" 0.4.0
[INFO   ] [deps        ] Successfully imported "kivy_deps.glew" 0.3.1
[INFO   ] [deps        ] Successfully imported "kivy_deps.sdl2" 0.7.0
[INFO   ] [Kivy        ] v2.3.0
[INFO   ] [Kivy        ] Installed at "C:\Users\AYUSHMAN\AppData\Roaming\Python\Python312\site-packages\kivy\__init__.py"
[INFO   ] [Python      ] v3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)]
[INFO   ] [Python      ] Interpreter at "C:\Python312\python.exe"
[INFO   ] [Logger      ] Purge log fired. Processing...
[INFO   ] [Logger      ] Purge finished!
[INFO   ] [Factory     ] 195 symbols loaded
[INFO   ] [Image       ] Providers: img_tex, img_dds, img_sdl2 (img_pil, img_ffpyplayer ignored)
[INFO   ] [Text        ] Provider: sdl2
[INFO   ] [Window      ] Provider: sdl2
[INFO   ] [GL          ] Using the "OpenGL" graphics system
[INFO   ] [GL          ] GLEW initialization succeeded
[INFO   ] [GL          ] Backend used <glew>
[INFO   ] [GL          ] OpenGL version <b'4.6.13596 Compatibility Profile Context 20.10.28.10 27.20.11028.10001'>
[INFO   ] [GL          ] OpenGL vendor <b'ATI Technologies Inc.'>
[INFO   ] [GL          ] OpenGL renderer <b'AMD Radeon(TM) Graphics'>
[INFO   ] [GL          ] OpenGL parsed version: 4, 6
[INFO   ] [GL          ] Shading version <b'4.60'>
[INFO   ] [GL          ] Texture max size <16384>
[INFO   ] [GL          ] Texture max units <32>
[INFO   ] [Window      ] auto add sdl2 input provider
[INFO   ] [Window      ] virtual keyboard not allowed, single mode, not docked
[INFO   ] [Base        ] Start application main loop
[INFO   ] [GL          ] NPOT texture support is available
[INFO   ] [Blank PCAPNG file 'C]\Users\AYUSHMAN\OneDrive\Desktop\test\Data.pcapng' created
[INFO   ] Capturing Data...
Capturing on 'Wi-Fi'
470 |
```



```
Blank PCAPNG file 'Data.pcapng' created.
File: %5cPublic%5c2023-04-25 22-38-06.mkv
No of Antivirus Software Searched through: 60
No of Suspected Virus Found: 0
```

As shown in the **Figure** above when a file is sent over a P2P connection from one computer to other, the file is checked for virus by the model made in this study. The results are displayed on the GUI created. If any virus will be detected it will be displayed are as threat and the file will be deleted immediately. This way the viruses coming along with the files and virus files will be eliminated. This keep the system secure and protected from any kind of virus attack over a P2P connection.

FUTURE WORK

Here are some potential directions for future work in this model. Developing more sophisticated techniques to identify and block malicious P2P traffic, including advanced malware, phishing attempts, and unauthorized data transfers. This could involve incorporating machine learning algorithms, behavioral analysis, and real-time threat intelligence feeds.

Implementing mechanisms to dynamically generate and adapt firewall rules based on real-time network traffic patterns and threat intelligence. This would enable the firewall to respond proactively to evolving threats and protect against zero-day vulnerabilities. Exploring ways to integrate the firewall with popular P2P network protocols, such as BitTorrent and Gnutella, to enhance security and performance. This could involve developing custom protocol extensions or utilizing existing security mechanisms within the protocols. Optimize the firewall's performance to handle large volumes of P2P traffic without compromising effectiveness. Ensure scalability to support growing network deployments and increasing user demands.

CONCLUSION

In conclusion, this project demonstrates the development of a robust and versatile Python-based program designed to significantly enhance network security and threat analysis within Peer-to-Peer (P2P) communication environments. The program leverages Wireshark for intelligent network traffic capture and integrates the VirusTotal API for comprehensive file analysis. This combined approach offers several key benefits:

Granular Control and Secure Data Management: The program facilitates fine-grained control over data capture through customizable filters, ensuring efficient extraction of specific P2P protocols. Captured data is securely stored in PCAPNG format and centralized via SMB shares, enabling streamlined access and historical analysis.

Enhanced Threat Detection with VirusTotal Integration: Seamless integration with the VirusTotal API empowers the program to scan captured files against a vast malware signature database, significantly bolstering its ability to identify both established and emerging threats. This real-time analysis empowers network administrators to take swift action in mitigating potential security risks.

Actionable Alerts and Comprehensive Logging: In response to security incidents, the program generates detailed alerts that provide critical insights into the nature, source, and impact of detected threats. Additionally, meticulous logs are maintained for all network activities, data captures, and analysis results, ensuring comprehensive oversight and facilitating auditing and forensic analysis.

User-Centric Design for Diverse Environments: Recognizing the unique needs of network environments, the program offers a high degree of user configurability. This allows network administrators to tailor capture filters, define export locations, and adjust virus detection sensitivity, aligning the program's behavior with their specific security posture.

By providing these functionalities, this Python program empowers network security professionals to gain deep visibility into P2P network activity, proactively manage traffic, and effectively safeguard their networks from evolving threats. This comprehensive solution offers significant value for organizations seeking to maintain a secure and well-controlled P2P network environment.