Detection of TrickBot and Emotet Banking Trojans with Machine Learning

Ruveyda Celik and Ali Gezer

Abstract— **Internet banking is getting more popular with the increasing number and demand of online banking customers. Almost all transactions that could be performed in bank branches could also be realized through internet banking. Internet banking, which has become widespread with the increasing use of the Internet, has also led to an increase in cases of financial fraud. This has made the protection of personal data and the security of banking services more important than ever. It is very important for institutions and organizations providing online banking services to take security measures in their systems. Cybercriminals target internet users with methods such as malware infection, botnets, spam, phishing, identity theft, and social engineering that they use and develop every day. Therefore, there are always potential risks in using internet banking. Banking viruses commonly used by cybercriminals today are TrickBot and Emotet. Nowadays TrickBot and Emotet are popular banking trojans which gives hard times for onlıne banking customers. Their primary goal is to steal user's banking and personal information. In this study, we will investigate the behavior analysis and new tricks of TrickBot and Emotet banking viruses, which use different methods to compromise the security of online banking customers. We benefited WEKA program to detect these banking viruses. In addition to this, we also focused on the detection of TrickBot and Emotet Banking viruses with using Random Tree, J48, Naive Bayes, SMO Techniques.**

Index Terms— ***Banking Trojan, Emotet, Machine Learning Methods, Malware Analysis, TrickBot, Web Injections***

I. Introduction

BANKING TROJANS are viruses that pretend to be a legitimate program or file, infiltrate computers and perform harmful actions. Although no one wants to be exposed to cyberattacks, millions of people become victim of attackers each year. In addition, banking viruses can create a backdoor that can copy the credentials of a bank customer by imitating the login web page of financial institutions.

TrickBot and Emotet are popular banking trojans that make such transactions from online banking and finance sites to attackers digital systems. Emotet is a Trojan that is primarily spread through spam emails (malspam). The malware may infect either via malicious script, macro-enabled document files, or malicious links. Emotet emails may contain familiar branding which designed to look like a legitimate email. Emotet may try to persuade users to click the malicious files by using tempting language such as “Your Invoice,” “Payment Details,” or possibly an upcoming shipment from well-known parcel companies [1].

Emotet has gone through a few iterations. Early versions arrived as malicious JavaScript file. Later versions evolved to use macro-enabled documents to retrieve the virus payload from command and control (C&C) servers.

Emotet uses a number of tricks to try and prevent detection and analysis. Notably, Emotet knows if it’s running inside a virtual machine (VM) and will lay dormant if it detects a sandbox environment, which is a tool cybersecurity researchers use to observe malware within a safe, controlled space [2].

Emotet also uses C&C servers to receive updates. This works in the same way as the operating system updates on your PC and can happen seamlessly and without any outward signs. This allows the attackers to install updated versions of the software, install additional malware such as other banking trojans, or to act as a dumping ground for stolen information such as financial credentials, usernames, passwords, and email addresses [3].

TrickBot was created to steal users' banking information. When Malwarebytes researchers first discovered TrickBot in 2016, they thought it was an ordinary identity theft purpose malware. But TrickBot targeted financial services and users for their banking data. It has also exploited other malwares to achieve its goals [4].TrickBot has the reputation of being the successor to another credential thief, Dyreza, who first appeared in 2014. TrickBot shared similarities with Dyreza, such as certain variables with similar values and the way that the functioning of command and control (C&C) servers. This has led many researchers to believe that the person or group that created Dyreza also created TrickBot [5].

In 2017, developers added a worm module to TrickBot, which we believe was inspired by successful ransomware campaigns with worm-like capabilities such as WannaCry and EternalPetya [6]. The developers also added a module for collecting Outlook credentials. The reason for adding this module is that hundreds of organizations and millions of people around the world often use this web mail service. The range of data TrickBot plays has also expanded. These are: cookies, browsing history, URLs visited, Flash LSO (Local Shared Objects) and many more. Although these modules were new at that time, they weren’t coded well enough.

In 2018, TrickBot continued to exploit the SMB

RÜVEYDA ÇELİK, is with Department of Electrical and Electronics Engineering University of Kayseri University, Kayseri, Turkey, (e-mail: ruveydacelik38@hotmail.com).

https://orcid.org/0000-0003-4821-4633

ALİ GEZER, is with Department of Cyber Security Application and Research Center University of Kayseri, Kayseri, Turkey, (e-mail: agezer@kayseri.edu.tr).

https://orcid.org/0000-0001-8265-1736

Manuscript received Dec 1, 2021; accepted Jul 4, 2022.
DOI: 10.17694/bajece.1031021

vulnerability. It was also equipped with the module that disables Windows Defender's real-time monitoring using a PowerShell command. While it had also updated its encryption algorithm, the rest of its module function stayed the same. TrickBot developers also started securing their code from being taken apart by security researchers via incorporating obfuscation elements [7]. At the end of the year, TrickBot was ranked as the top threat against businesses, and has overtaken Emotet. TrickBot developers made some changes to the Trojan in 2019. Specifically, they made changes to the way that webinject feature works against the some US-based mobile carriers.

Recently, researchers have noted an improvement in this Trojan's evasion method. Mworm, the module responsible for spreading a copy of itself, was replaced by a new module called Nworm. This new module alters TrickBot's HTTP traffic, allowing it to run from memory after infecting a domain controller. This ensures that TrickBot doesn't leave any traces of infection on affected machines.

These banking viruses have allowed them to install updated versions of the software, install additional malware such as other banking Trojans, or act as intermediaries for stolen information such as financial credentials, usernames, passwords, and email addresses [8]. The chain of infection diagram for banking malicious viruses for Emotet and TrickBot is shown in Figure 1.

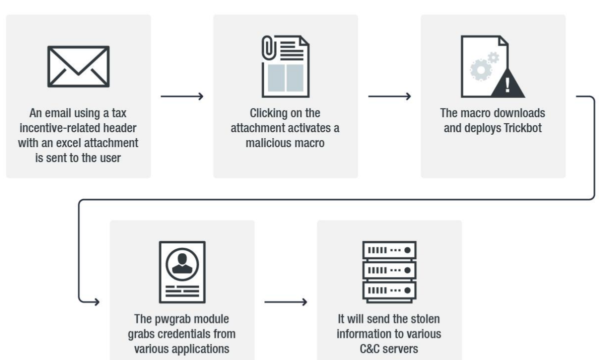


Fig.1. Infection chain for Emotet and TrickBot

II. PROSED METHOD

A. Data Collection Through Static And Dynamic Analysis

Static and dynamic analyzes could be performed to reveal the signatures of malicious softwares. Via determining network flows, virus detection was investigated in virtual machines infected with TrickBot and Emotet. When the TrickBot and Emotet trojans infect a system, their first action is to identify their victims. It performs a network activity via e-mail or HTTP request to some service sites, websites of targeted banks or websites where users can access their personal data.

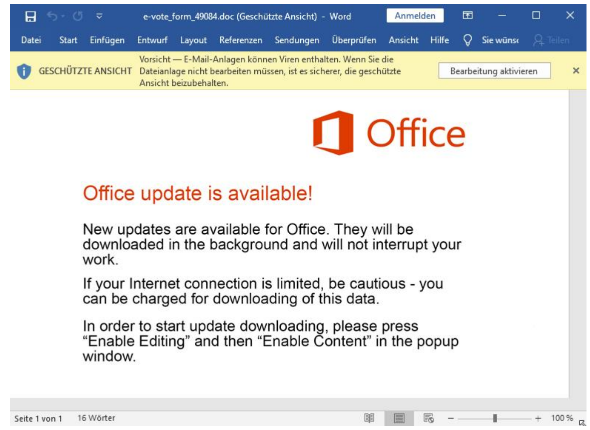


Fig.2. Suspicious e-mail content

When the infected file in the email is executed the compromised system tries to connect to one of Command and Control servers of these two currently active banking viruses. Some server IPs are encoded into the malware's binary. After the connection is established, TrickBot and Emotet viruses try to download the encrypted file modules. They try to access new IPs by leaking stolen data to Downloaders. We can see the IPs that these viruses interact with using Process Hacker and Wireshark programs.

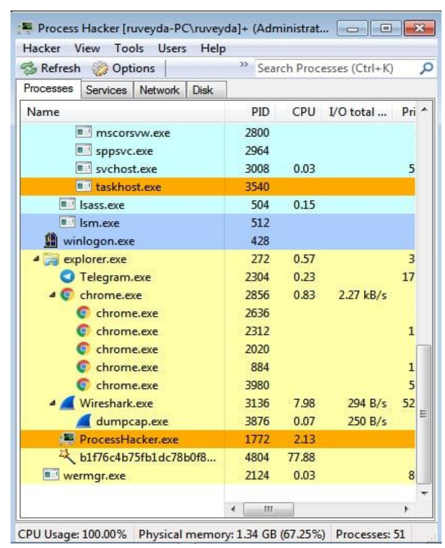


Fig.3. Process Hacker output of a computer infected with Emotet

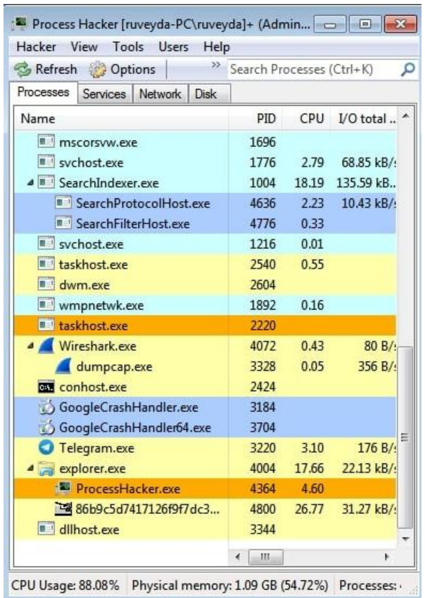


Fig.4. Process Hacker output of a computer infected with TrickBot

Process Hacker is an open source tool that allows you to see what processes are running on a device, identify programs consuming CPU resources, and identify network connections associated with a process. Such features make Process Hacker an ideal tool for monitoring malware on a device. Being able to see what processes are spawned identify network connections and interesting threads could give us valuable indicators of danger (IOCs) [9].

IP addresses and malicious domain names are valuable indicators in incident response. Using Process Hacker is helpful to gather such information which also compromised hosts can be identified in the network.

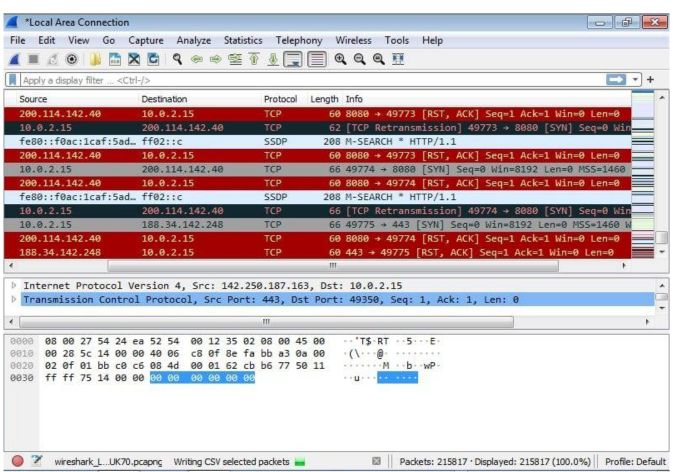


Fig.5. Wireshark output of a computer infected with TrickBot

Wireshark is a packet sniffer and protocol analysis tool. It captures network traffic in the local network and stores this data for offline analysis. Wireshark captures network traffic from Ethernet, Bluetooth, Wireless (IEEE.802.11), Token Ring, Frame Relay connections and more. Wireshark lets you filter the log before capture starts or during analysis so you can narrow down what you're looking for. For example, you can set a filter to see TCP traffic between two IP addresses. You can set it to show you only packets sent from a computer. The powerful filtering mechanisms in Wireshark is one of the main reasons it has become the standard tool for packet analysis [10].

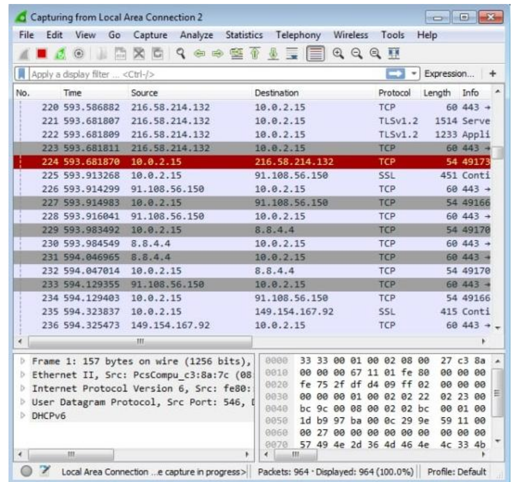


Fig.6. Wireshark output of a computer infected with Emotet

When we look at network traffic, the compromised system tries to make a connection to one of the C&C servers of TrickBot and Emotet. Some server IPs are encoded into the malware's binary. After the connection is established, TrickBot and Emotet viruses try to download the encrypted file modules. It tries to access new IP's by leaking stolen data.

TrickBot and Emotet virus download files in AppData folder. While TrickBot spawns in Roaming, Emotet spawns in Local directory.

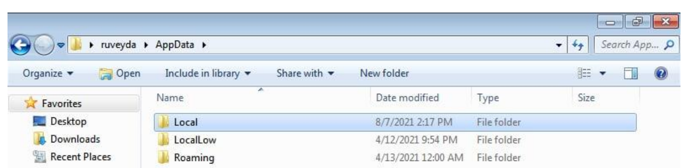


Fig.7. Directories that TrickBot and Emotet cloned themselves

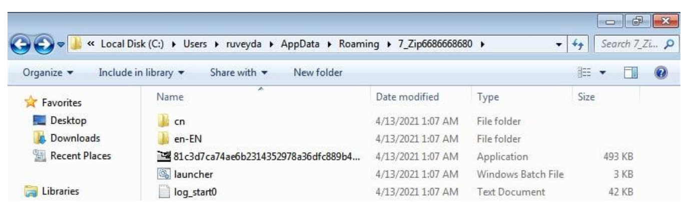


Figure.8. Created Files after TrickBot infection

TrickBot cloned itself in C:\Users*\AppData\Roaming\7_Zip6686668680) copied itself and downloaded different files for different purposes.

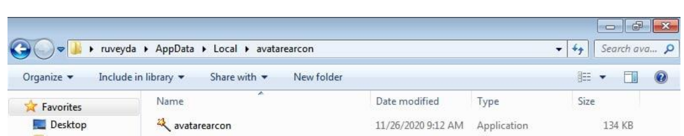


Figure.9. Created Files after Emotet Infection

Emotet cloned itself in

C:\Users*\AppData\Local\avatarearcom and downloaded files to realized its purposes.

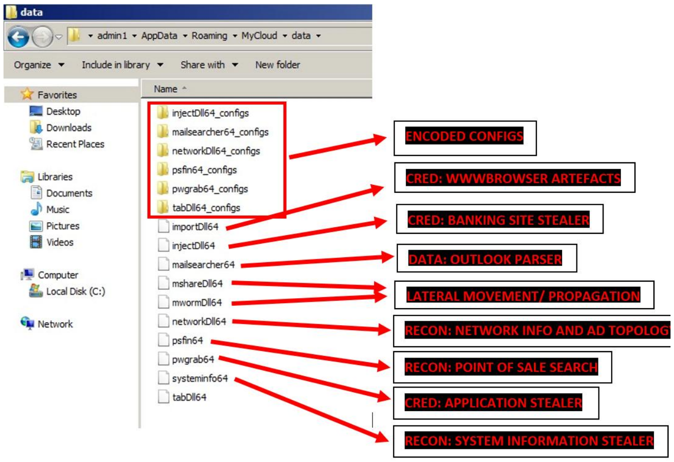


Fig.10. Created files and downloads by Emotet and TrickBot infection

Emotet and TrickBot download files for some specific purposes. Each of these files has different properties and purposes. The most important of these are as follows:

▪ TrickBot and Emotet modules are delivered as Dynamic Link Libraries (DLLs) loaders.

▪ Mainly TrickBot and Emotet have two core modules, Injectdll and systeminfo.

▪ Injectdll module is used to target banking and financial data, monitors banking website activity and uses web injections to steal financial information. Systeminfo is used to fingerprint the infected system specifications [11].

▪ Besides the above features, other files TrickBot and Emotet download are as follows:

- **ModuleDll/ImportDll:** Collects browser data (eg cookies and browser configurations).
- **Dinj:** File contains bank information; uses server-side web injections.
- **Dpost:** Most of the data leaked by TrickBot is sent to the dpost IP address.
- **Sinj:** Keeps information about targeted online banks; Uses redirect attacks (fake web injections) to leak financial data
- **DomainDll:** Uses LDAP to collect credentials and configuration data. Domain controller by accessing shared SYSVOL files.
- **OutlookDll:** Harvests saved Microsoft Outlook credentials by querying several registry keys.
- **SqulDll:** Force enables WDigest authentication and utilizes Mimikatz to scrape credentials from LSASS.exe. The worming modules use these credentials to spread TrickBot and Emotet laterally across networks.
- **NetworkDll, wormDll, shareDll:** Used for network reconnaissance and lateral movement.
- **RdpScanDll:** Bruteforces RDP for a specific list of victims.

TrickBot banking trojan uses a domain creation algorithm to communicate with its servers. Once infected, the trojan starts executing DNS queries for the created domains. Another popular banking trojan, Emotet, which exhibits a completely different network communication model, sets up a local proxy and routes internet traffic through the proxy server.

Modules can be downloaded from one of TrickBot's or Emotet's C2s using simple GET requests such as https://<CC_IP>:<CC_PORT>/<gtag>/<bot_ID>/5/<module_name>/. Although module names are case sensitive and we define 32-bit modules, in most cases 64-bit versions can be downloaded by typing '64' instead of '32' in the module name. In most cases, valid <gtag> and <bot_ID> values are not required for successful download. Files which are encrypted could be decrypted with the following Python script [12].

TABLE I
PROPERTIES OF FILES DOWNLOADED TICKBOT AND EMOTET

Name	Function
importDll64	Browser data stealer module
injectDll64	Handles web-injects, including support for several hundred banking/financial sites
mailsearcher64	Recon module parses specific file types for “of interest” data
mshareDll64	Lateral movement / enumeration module via LDAP and SMB exploitation. Mshare and mworm modules work in cooperation
mwormDll64 mshareDll	Lateral movement / enumeration module via LDAP and SMB exploitation. Mshare and mworm modules work in cooperation
networkDll64	Recon module queries network specific environmental data
psfin64	Point-of-sale recon module
pwgrab64	Steals credentials, autofill data, history, and other information from browsers as well as several software applications.
systeminfo64	Recon module. Provides system-specific information and data to the C2
tabDll64	Credential theft module. Sometimes contains additional lateral movement code. Uses the EternalRomance exploit (CVE-2017-0147) to spread via SMBv1.

```python
import hashlib
from Crypto.Cipher import AES

def hash_rounds(data):
    while len(data) <= 0x1000:
        buf_hash = hashlib.sha256(data).digest()
        data += buf_hash
    return buf_hash

def decrypt(data):
    pad = lambda s: s + (16 - len(s) % 16) * chr(16 - len(s) % 16)
    key = hash_rounds(data[:0x20])[:0x20]
    iv  = hash_rounds(data[0x10:0x30])[:0x10]

    aes = AES.new(key, AES.MODE_CBC, iv)
    data = pad(data[0x30:])
    return aes.decrypt(data)
```

Fig.11. Python script with module decrypt routine

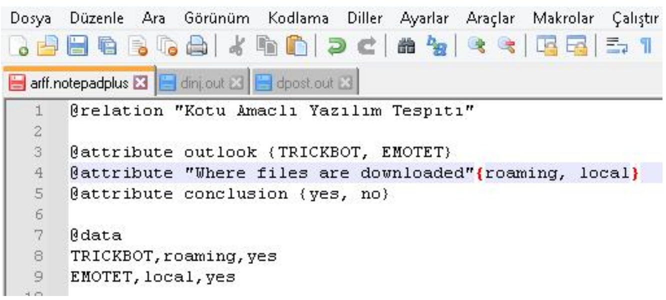


Fig.12. Notepadplus arff file created as a result of Python password analysis

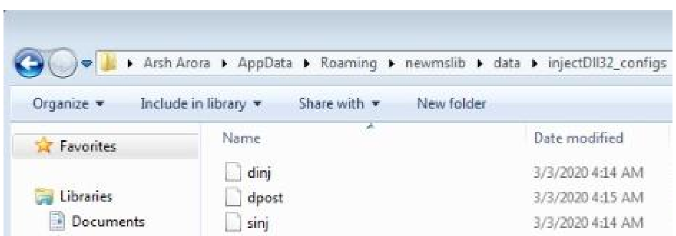


Fig.13. Encrypted dinj, dpost and sinj files

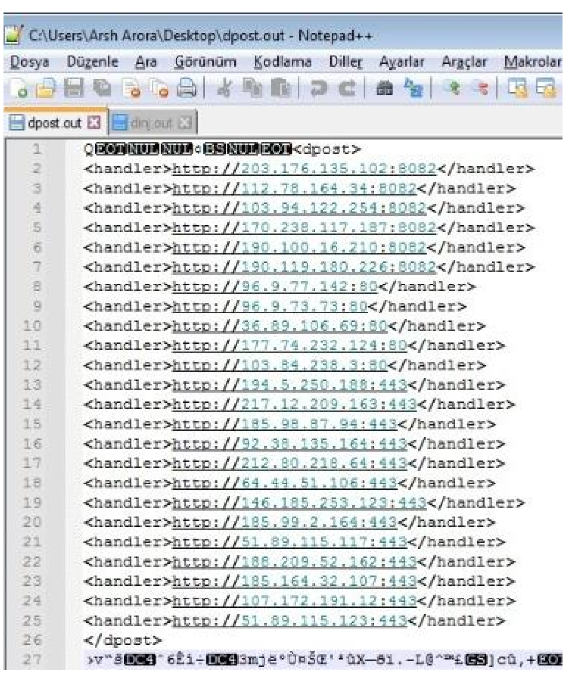


Fig.14. Decrypted dpost.out arff file with Python

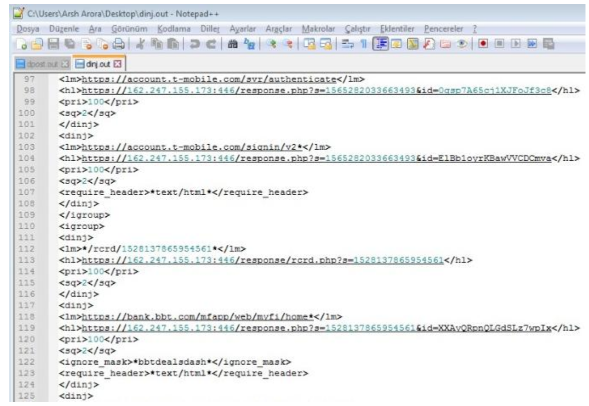


Fig.15. Decrypted dnj.out arff file with Python

We know that TrickBot and Emotet use a virtual network system that allows them to take over the victim's computer systems. TrickBot also used different modules to enter user credentials into any banking session. It mostly gets the downloaded modules and configuration files by running them on their servers. After running these modules, it also needs a network communication to accomplish its destructive goals. However, due to the encrypted content of the exchanged packets, their purpose is difficult to understand. Therefore, revealing the TrickBot and Emotet networking pattern will help us detect any viral infection in a system. We focus on the communication patterns between the TrickBot and Emotet servers and the compromised system.

TrickBot and Emotet network traffics were examined to determine network flow patterns. We will use machine learning techniques to detect the TrickBot and Emotet infection. Each traffic flow is defined by a set of statistical properties that can be calculated from one or more packages. Therefore, each stream will be characterized by the same set of attribute names, but different attribute values [13].

B. Machine Learning Approach to Identify TrickBot and Emotet Streams Color Space

In our methodology, we use a supervised machine learning approach to classify traffic flows by class membership. In supervised classification, classes must be predefined before the system is trained. First, a classification model is used via using a training dataset containing examples of each class. This model is then used to predict class membership for new traffic flows represented as statistical features.

We analyzed many TrickBot and Emotet malware samples statically and dynamically over 1 year period. After executing the samples of TrickBot and Emotet, a network is created between the compromised computer and a URL call is performed to reveal the public IP address of the infected computer. During the dynamic analysis, we observed the %AppData%Roaming folder to see if there were any newly created folders related to the TrickBot infection. We observed the %AppData%Local folder to see if there were any newly created folders related to the Emotet infection [14].

Before running TrickBot and Emotet malware samples in a virtual machine, we set it up to capture network traffic with some predefined filtering rules to not to capture broadcast packets with the Wireshark protocol analyzer. In Windows Task Manager it can be easily observed when the TrickBot and

Emotet process is started, finished and deployed and how many svchost processes are started by the main executable to run the downloaded module DLLs [15].

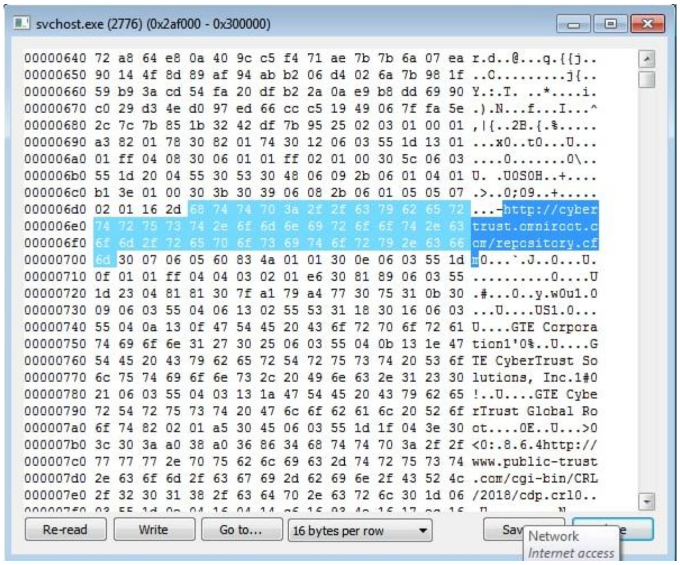


Fig.16. ASCII codes that appear in svchost after TrickBot and Emotet viruses are executed

Svchost.exe actually stands for "service host" and it is a file used by most Windows applications. Despite this, it is generally considered a virus, as malware developers are known to add malicious files to the svchost.exe service to avoid detection. In addition, malware authors often create misspelled files such as "svhost.exe" and "svchosl.exe" to avoid detection by observers.

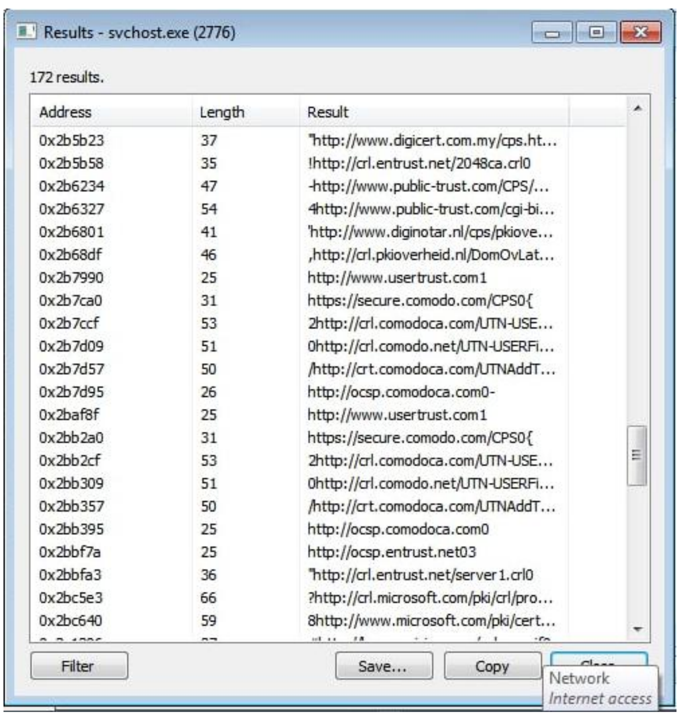


Fig.17. Website samples obtained as a result of svchost filtering after infection

We analyzed more than 100 different TrickBot and Emotet samples over 1 year to observe how TrickBot and Emotet banking viruses evolve and discover new behavioral patterns.

With Static and Dynamic analysis, we observed the interaction of TrickBot and Emotet on our computer after they were executed We filtered the network traffic through the Wireshark protocol analyzer. We perform this because we define a network communication model and show that this interruption is caused by TrickBot and Emotet related flows in the network. During the process of capturing network traffic while the TrickBot instance is running, initial HTTP traffic is intentionally generated through interaction with popular domains. Such bona fide traffic is generated by visiting university domains, online newspaper, well-known social media websites, and some well-known websites [16].

The .pcap files captured this way also contain regular web traffic, as opposed to containing only TrickBot and Emotet-specific traffic. That's why we run each sample (infected file content) at different times. Sometimes when visiting banking web sites, it may take only a few minutes initially to observe the network traffic, and sometimes more than 2 hours to observe the injection. We captured pcap files containing both TrickBot and Emotet related traffic and also benign traffic. This difference in traffic captured in .pcap files is very useful for training and testing data for our proposed model [17]. The QUIC protocol is often observed after the Emotet virus is executed.

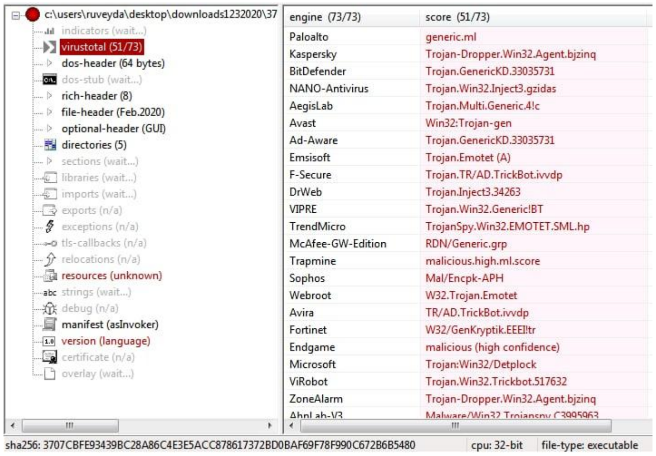
QUIC is an experimental low-latency new internet protocol implemented by Google over UDP. Generally, UDP is used in areas where speed is important and latency is not tolerated.QUIC is a protocol developed by Google. QUIC supports replicated link aggregation and aims to provide secure data transmission with similar features to TLS/SSL. It works in the same structure as the HTTP/2 protocol, but contains features that the HTTP/2 protocol cannot provide [18].

QUIC has taken a new approach to reduce latency by addressing the problems of packet loss and long RTTs (Round-round Times). It manages the former using ubiquitous TCP with UDP (User Datagram Protocol) and then minimizes the number of round trips between the sender and receiver. TCP-based delays on websites sometimes exceed milliseconds and reach up to seconds. This is where Google's new protocol QUIC comes into play. For this reason, Emotet generally prefers to exchange packets over this protocol [19].

III. IMPLEMENTATION

It is very difficult to reveal the signatures of banking viruses, which can be transmitted via e-transmitted during any banking transaction. The methodology we developed here is related to the detection of banking viruses that are harmful in such cases. While creating our classification model, we defined a data set in Excel environment to summarize different characteristics. While creating our dataset, we benefited from traffic flows and also HTTP adresses in svchost.

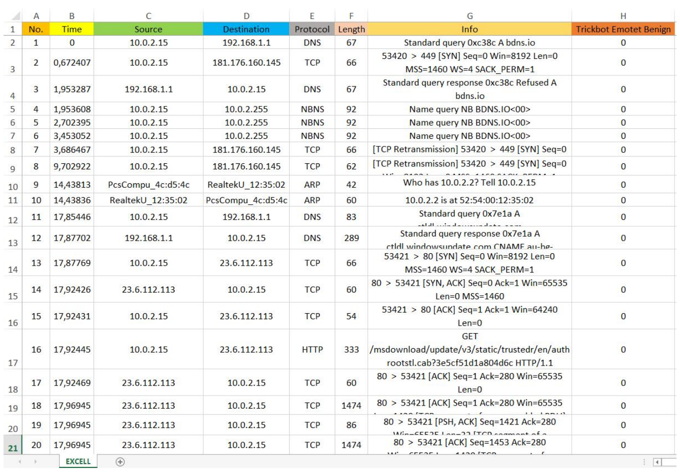
Our data set was created by examining the protocols. During the dataset collection phase, we collected 41437 samples from different sources including Contagio security block, MalDozer, VirusTotal, AMD datasets.



engine (73/73)	score (51/73)
Paloalto	generic.ml
Kaspersky	Trojan-Dropper.Win32.Agent.bjzinq
BitDefender	Trojan.GenericKD.33035731
NANO-Antivirus	Trojan.Win32.Inject3.gzidas
AegisLab	Trojan.Multi.Generic.4!c
Avast	Win32:Trojan-gen
Ad-Aware	Trojan.GenericKD.33035731
Emsisoft	Trojan.Emotet (A)
F-Secure	Trojan.TR/AD.TrickBot.ivvdp
DrWeb	Trojan.Inject3.34263
VIPRE	Trojan.Win32.Generic!BT
TrendMicro	TrojanSpy.Win32.EMOTET.SML.hp
McAfee-GW-Edition	RDN/Generic.grp
Trapmine	malicious.high.ml.score
Sophos	Mal/Encpk-APH
Webroot	W32.Trojan.Emotet
Avira	TR/AD.TrickBot.ivvdp
Fortinet	W32/GenKryptik.EEEI!tr
Endgame	malicious (high confidence)
Microsoft	Trojan:Win32/Detplock
ViRobot	Trojan.Win32.Trickbot.517632
ZoneAlarm	Trojan-Dropper.Win32.Agent.bjzinq

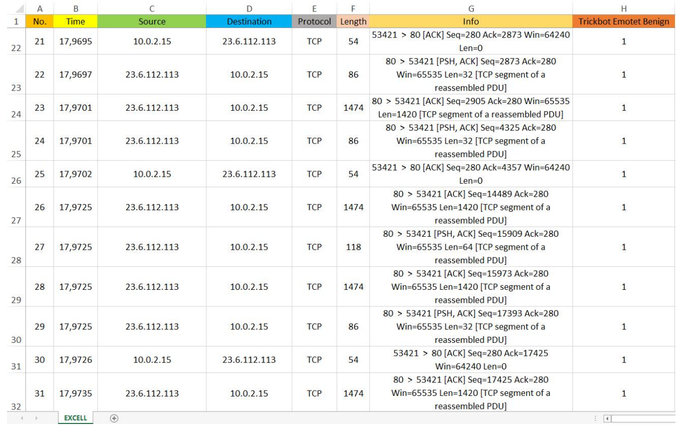
Fig.18. Infected IPs discovered as a result of Vırustotal analysis

Due to the analysis performed, 13077 out of 41437 samples were determined as working samples. After data collection from December 2021 to December 2022, we categorized the data according to its functionality whether it was malware and, if so, what type. As a result, we obtained a set containing 13077 data, consisting of a total of 5 categories. Our dataset has 9803 row, including Benign flows, Banking Malware (Emotet, TrickBot). The row number of benign flows are 1795 and it is marked as in the benign software category. They also presented as two different sets which contain 470 and 139 features. In this study, datasets containing 500 extracted features were used. Below, we display the data set, which contain 10 Benign, 10 TrickBot and 10 Emotet samples.



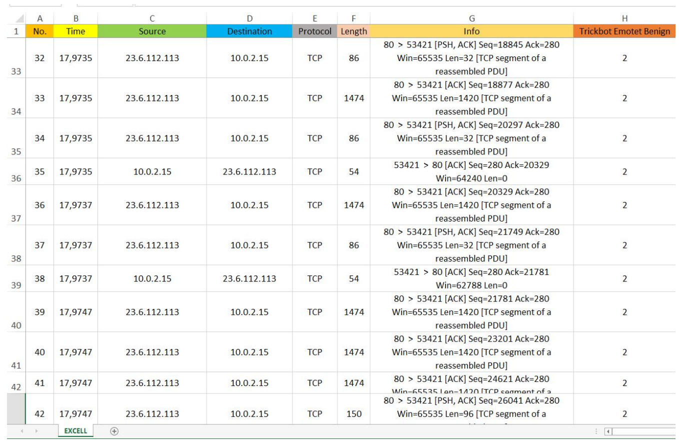
No.	Time	Source	Destination	Protocol	Length	Info	Trickbot Emotet Benign
1	0	10.0.2.15	192.168.1.1	DNS	67	Standard query 0xc38c A bdns.io	0
2	0,672407	10.0.2.15	181.176.160.145	TCP	66	53420 > 449 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	0
3	1,953287	192.168.1.1	10.0.2.15	DNS	67	Standard query response 0xc38c Refused A bdns.io	0
4	1,953608	10.0.2.15	10.0.2.255	NBNS	92	Name query NB BDNS.IO<00>	0
5	2,702395	10.0.2.15	10.0.2.255	NBNS	92	Name query NB BDNS.IO<00>	0
6	3,453052	10.0.2.15	10.0.2.255	NBNS	92	Name query NB BDNS.IO<00>	0
7	3,686467	10.0.2.15	181.176.160.145	TCP	66	[TCP Retransmission] 53420 > 449 [SYN] Seq=0	0
8	9,702922	10.0.2.15	181.176.160.145	TCP	62	[TCP Retransmission] 53420 > 449 [SYN] Seq=0	0
9	14,43813	PcsCompu_4c:d5:4c	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15	0
10	14,43836	RealtekU_12:35:02	PcsCompu_4c:d5:4c	ARP	60	10.0.2.2 is at 52:54:00:12:35:02	0
11	17,85446	10.0.2.15	192.168.1.1	DNS	83	Standard query 0x7e1a A	0
12	17,87702	192.168.1.1	10.0.2.15	DNS	289	Standard query response 0x7e1a A	0
13	17,87769	10.0.2.15	23.6.112.113	TCP	66	53421 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	0
14	17,92426	23.6.112.113	10.0.2.15	TCP	60	80 > 53421 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460	0
15	17,92431	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0	0
16	17,92445	10.0.2.15	23.6.112.113	HTTP	333	GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?3e5cf51d1a804d6c HTTP/1.1	0
17	17,92469	23.6.112.113	10.0.2.15	TCP	60	80 > 53421 [ACK] Seq=1 Ack=280 Win=65535 Len=0	0
18	17,96945	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=1 Ack=280 Win=65535	0
19	17,96945	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=1421 Ack=280	0
20	17,96945	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=1453 Ack=280	0

Fig.19.a.Dataset example Benign flows (except TrickBot and Emotet)



No.	Time	Source	Destination	Protocol	Length	Info	Trickbot Emotet Benign
21	17,9695	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=280 Ack=2873 Win=64240 Len=0	1
22	17,9697	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=2873 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	1
23	17,9701	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=2905 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	1
24	17,9701	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=4325 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	1
25	17,9702	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=280 Ack=4357 Win=64240 Len=0	1
26	17,9725	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=14489 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	1
27	17,9725	23.6.112.113	10.0.2.15	TCP	118	80 > 53421 [PSH, ACK] Seq=15909 Ack=280 Win=65535 Len=64 [TCP segment of a reassembled PDU]	1
28	17,9725	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=15973 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	1
29	17,9725	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=17393 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	1
30	17,9726	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=280 Ack=17425 Win=64240 Len=0	1
31	17,9735	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=17425 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	1

Fig.19.b.Dataset example for TrickBot



No.	Time	Source	Destination	Protocol	Length	Info	Trickbot Emotet Benign
32	17,9735	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=18845 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	2
33	17,9735	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=18877 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	2
34	17,9735	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=20297 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	2
35	17,9735	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=280 Ack=20329 Win=64240 Len=0	2
36	17,9737	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=20329 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	2
37	17,9737	23.6.112.113	10.0.2.15	TCP	86	80 > 53421 [PSH, ACK] Seq=21749 Ack=280 Win=65535 Len=32 [TCP segment of a reassembled PDU]	2
38	17,9737	10.0.2.15	23.6.112.113	TCP	54	53421 > 80 [ACK] Seq=280 Ack=21781 Win=62788 Len=0	2
39	17,9747	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=21781 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	2
40	17,9747	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=23201 Ack=280 Win=65535 Len=1420 [TCP segment of a reassembled PDU]	2
41	17,9747	23.6.112.113	10.0.2.15	TCP	1474	80 > 53421 [ACK] Seq=24621 Ack=280	2
42	17,9747	23.6.112.113	10.0.2.15	TCP	150	80 > 53421 [PSH, ACK] Seq=26041 Ack=280 Win=65535 Len=96 [TCP segment of a	2

Fig.19.c.Dataset example for Emotet

The analysis of the dataset using machine-learning classifiers was carried out with the WEKA software which was developed at the University of Waikato. It is the abbreviation for Waikato Environment for Knowledge Analysis. This code, which is a JAVA open source library, contains an algorithm that can be applied to devices with Android operating system [20]. In the classification results made with WEKA, False Positive Ratio (FPR), True Positive Ratio (TPR), Precision, Recall, FMeasure etc. values are given. These values are an important criterion in interpreting the results. TPR, correctly defined data; FPR, misidentified data; Precision is expressed as the ratio of the correct data of a category to the incorrect data of that category and is formulated as follows [20, 21].

$$TPR = TP\ FN+TP \tag{1}$$

$$FPR = FP\ TN+FP \tag{2}$$

$$Precision = TP\ TP+FP \tag{3}$$

$$F-Measure = 2*TP\ 2*TP+FP+FN \tag{4}$$

$$Recall = TP\ FN+TP \tag{5}$$

$$Accuracy(\%) = TN+TP\ TP+FN+FP+TN * 100 \tag{6}$$

The path followed in the study is given in Figure 20. Up to this point, the dataset and WEKA evaluation criterias are included. The next steps are covered in the 'Results and Discussion' section in detail.

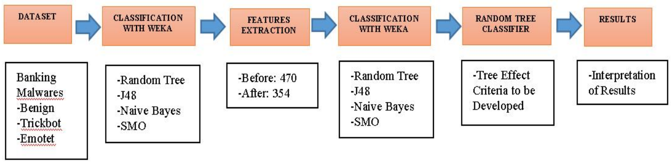


Fig.20. Flow chart of study

In this study, in which analyzes were made for the detection of malware, first of all, the data set was first analyzed using machine learning (ML) classifiers such as SMO, Naive Bayes (NB), J48 and Random Tree (RT) algorithms. Then, feature extraction was performed and the results were compared with the same ML classifiers. The effect of different parameters was examined by using the algorithm that gave the best results.

A. Effects of Algorithms

When the literature is examined, it is seen that ML (Machine Learning) algorithms are frequently used in malware detection. In this context, 4 different classifiers, namely SMO, NB (Naive Bayes), J48 and RT (Random Tree), were included in the study. Classification results using the WEKA program are given in Figures 21, 22, 23 and 24. It is seen that the Random Tree classifier is the algorithm that gives the best result with a success rate of 83% in the success evaluation using the accuracy percentage. The lowest success is the SMO algorithm with 60%. Results for J48 and NB were 77% and 64%, respectively. TPR, FPR etc. given in WEKA analysis outputs in Figure 25. The results of the criteria are given. All classifiers, SMS Malware appear to have high accuracy. This result is consistent with the findings from the study [22].

trees.RandomTree

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          83               83      %
Kappa statistic                          0.6072
Mean absolute error                      0.2785
Root mean squared error                  0.3582
Relative absolute error                 61.8292 %
Root relative squared error             75.5279 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,909    0,324    0,845      0,909    0,876      0,611    0,897     0,950     c0
                 0,676    0,091    0,793      0,676    0,730      0,611    0,897     0,790     c1
Weighted Avg.    0,830    0,244    0,827      0,830    0,826      0,611    0,897     0,896

=== Confusion Matrix ===

  a  b   <-- classified as
 60  6 |  a = c0
 11 23 |  b = c1
```

Fig.21. Random Tree classification algorithm example result in WEKA application

trees.J48

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          77               77      %
Kappa statistic                          0.4838
Mean absolute error                      0.3108
Root mean squared error                  0.3921
Relative absolute error                 68.9937 %
Root relative squared error             82.6609 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,833    0,353    0,821      0,833    0,827      0,484    0,820     0,870     c0
                 0,647    0,167    0,667      0,647    0,657      0,484    0,820     0,660     c1
Weighted Avg.    0,770    0,290    0,768      0,770    0,769      0,484    0,820     0,799

=== Confusion Matrix ===

  a  b   <-- classified as
 55 11 |  a = c0
 12 22 |  b = c1
```

Fig.22. Trees.J48 classification algorithm example result in WEKA application

bayes.NaiveBayes

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          64               64      %
Kappa statistic                          0.2834
Mean absolute error                      0.36
Root mean squared error                  0.6
Relative absolute error                 72.0627 %
Root relative squared error            120.0079 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,577    0,292    0,682      0,577    0,625      0,287    0,643     0,613     Value1
                 0,708    0,423    0,607      0,708    0,654      0,287    0,643     0,570     Value2
Weighted Avg.    0,640    0,355    0,646      0,640    0,639      0,287    0,643     0,593

=== Confusion Matrix ===

  a  b   <-- classified as
 30 22 |  a = Value1
 14 34 |  b = Value2
```

Fig.23. Naive Bayes classification algorithm example result in WEKA application

functions.SMO

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          60               60      %
Kappa statistic                          0.1961
Mean absolute error                      0.4664
Root mean squared error                  0.5008
Relative absolute error                 93.3564 %
Root relative squared error            100.1666 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,654    0,458    0,607      0,654    0,630      0,197    0,610     0,614     Value1
                 0,542    0,346    0,591      0,542    0,565      0,197    0,610     0,565     Value2
Weighted Avg.    0,600    0,404    0,599      0,600    0,599      0,197    0,610     0,591

=== Confusion Matrix ===

  a  b   <-- classified as
 34 18 |  a = Value1
 22 26 |  b = Value2
```

Fig.24. SMO classification algorithm example result in WEKA application

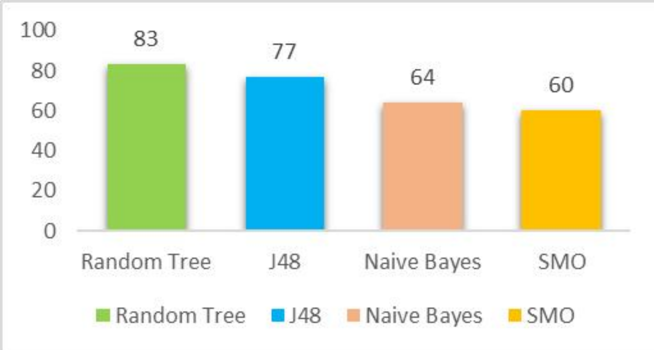


Fig.25. Classifier accuracy

	CATEGORY	TPR	FPR	PRECISION	RECALL	F-MEASURE	ACCURACY(%)
RANDOM TREE	1	0,909	0,324	0,845	0,909	0,876	83%
	2	0,676	0,091	0,793	0,676	0,73	
	3	0,83	0,244	0,827	0,83	0,826	
J48	1	0,833	0,353	0,821	0,833	0,827	77%
	2	0,647	0,167	0,667	0,647	0,657	
	3	0,77	0,29	0,768	0,77	0,769	
NAIVE BAYES	1	0,577	0,292	0,682	0,577	0,625	64%
	2	0,708	0,423	0,607	0,708	0,654	
	3	0,64	0,355	0,646	0,64	0,639	
SMO	1	0,654	0,458	0,607	0,654	0,63	60%
	2	0,542	0,346	0,591	0,542	0,565	
	3	0,6	0,404	0,599	0,6	0,599	

Fig.26. Classification optimization results from the WEKA

In Figure.26, the data numbers classified according to the categories are given. Category 1 (Benign), Category 2 (TrickBot), Category 3 (Emotet) are expressed as 1, 2, 3, respectively. Highest result (Random Tree- 83% vs. J48 - 77%), compared to the number of benign software detected in J48 (1459) is higher than that found in Random Tree (1379). Sum of numbers for each category data is 10607 for a correctly classified Random Tree. In J48, the correct classification result in all categories is as follows: 10181. In short, although the success of detecting benign software in J48 was 77% (83% for Random Tree), considering the overall percentage of accuracy, Random Tree appears to be a better classifier under these categories.

B. Feature Extraction Effect

Feature reduction is one of the key pieces of work in malware detection. In this study, 116 features with the lowest effect on the ranking were removed and reconstructed. We count 470 feature for the analysis. The results obtained for the RT, NB, J48 and SMO classifiers, compared with the results before feature extraction (Figure 27).

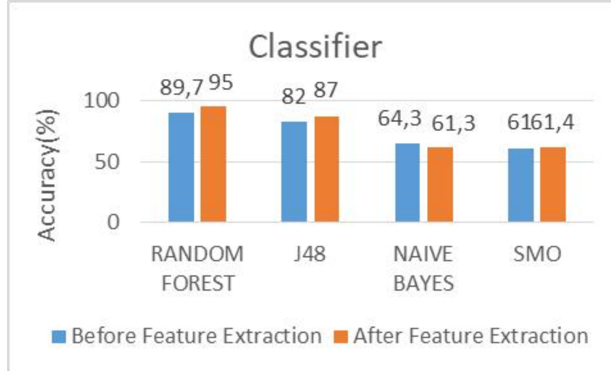


Fig.27. Accuracy comparison before and after feature extraction.

In Figure 27, the change in accuracy for the Random Tree classifier after feature extraction was minimal. The greatest increase in the accuracy of the results was observed for NB. Contrary to the others, there is a small decrease in J48. In the analysis made so far, the success of different classifiers and the effect of feature extraction in malware detection have been examined. In the comparison, as seen in Figure 25, the malware was tagged with the best Random Tree classifier. Based on this achievement, analyzes were made for the Random Tree classifier and 354 features in the next parts of the study.

C. Tree Effect Criteria to be Developed

According to the findings of the study, Random Tree is the best performing classifier for detecting banking malicious software (TrickBot and Emotet). Among other classifiers, new analyzes were made by changing the number based on this information.

Random forest algorithm in Random Tree algorithm is one of the supervised classification algorithms. It is used in both regression and classification problems. The algorithm aims to increase the classification value during the classification process by producing more than one decision tree. Random forest algorithm is the process of choosing the highest score among many decision trees that work independently of each other. As the number of trees (our data) increases, our rate of obtaining a precise result increases [23]. The main difference between the decision tree algorithm and the random forest algorithm is that the process of finding the root node and splitting the nodes is random. The random forest algorithm reduces the problem of over-learning if you have enough trees. It requires little data preparation. It requires little data preparation. It requires little data preparation. The aim is to observe ın the algorithm with the best classification, different parameters are tried to determine the accuracy and reach the final result.

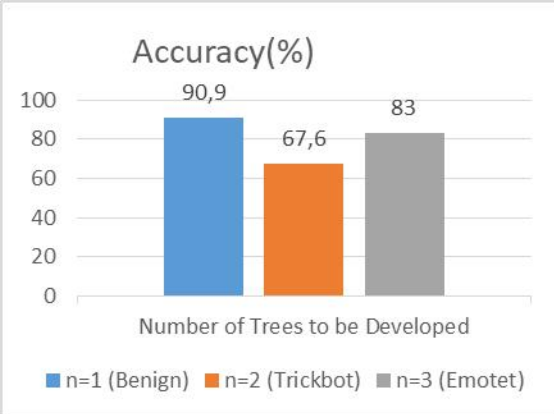


Fig.28. Tree Effect Criteria to be Developed

Figure 28 examines the effect of the number of trees to be developed for the Random Tree classifier. For Benign, Emotet and TrickBot an increase in n indicates the impact on malware detection.

It is concluded that for malware detection, the Random Tree classifier determines the best discrimination.

IV. CONCLUSION

The information age that we live in has brought along some problems as well as providing great convenience for humanity. As the access to information, technology and internet became easier, malicious use of internet also has become a significant problem. In parallel with the increase in these threats, which pose a great danger to information security, prevention and detection activities in these areas have also accelerated. In the field of information security, malware detection studies, which are also frequently encountered in the academic world, are conducted to identify threats developed with malicious intent. In this study, we develop a technique by using ML classifiers to determine the banking trojan infection.

The percentage of success in malware detection studies is associated with accurate detection of malwares. Tagging good software as malicious software can cost money and time. However, since labeling malware as benign will cause even greater damage. In this context, it has been seen that tagging malware correctly has improved the reliability of our study. Also in addition to the effect of feature extraction, we also study the classifier performance. According to the findings of these two phases, the best classification success (before and after feature extraction) belongs to Random Tree algorithm for TrickBot and Emotet detection. The change in the number of trees has provided the desired success in malware detection.

In our analysis, we observe that Random Tree and J48 give better results compared to other detection techniques. Despite higher flow detection with J48, Random Tree performed better overall. We obtained 83% Our dataset, which we ran in the Weka program, yielded the following results: Random Tree 83%, J48 77%, Naive Bayes 64% and SMO 60%.

In short, in this age where information is under threat, malware detection and prevention is of great importance. Detection of banking malicious software is one of the shining areas for the security of banking customers. This study has enriched the literature in terms of examining this correct labeling. After evaluating the classifier performance and feature extraction efficiency, Random Tree gives best results in terms of classification of benign TrickBot and Emotet traffic flows.

REFERENCES

[1] M. Edwin Agwu, "Analysis of Obstacles to Uptake of Internet Banking Services in Nigeria" Research Journal & Management-RJBM (2015), Vol.2(1)doi:10.17261/Pressacademia.201519824 Available: https://dergipark.org.tr/tr/download/article-file/375170

[2] M. Zainab Alkhalil, Chaminda Hewage "Phishing Attacks: A Recent Comprehensive Study and a New Anatomy" Liqaa Nawaf and Imtiaz Khan Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, United Kingdom Front. Comput. Sci., 09 March 2021 Available: https://www.frontiersin.org/articles/10.3389/fcomp.2021.563060/full

[3] Debbie Walkowski "Banking Trojans: A Reference Guide to the Malware Family Tree By Remi Cohen Additional Contributions" August 09, 2019 Available: https://www.f5.com/labs/articles/education/banking-trojans-a-reference-guide-to-the-malware-family-tree

[4] Cybersecurity and Infrastructure Security Team "Emotet Malware" July 20, 2018 Last Revised: January 23, 2020 Available: https://us-cert.cisa.gov/ncas/alerts/TA18-201A

[5] Michelle Drolet "What is Emotet? And how to guard against this Persistent Trojan Malware" Contributor, April 12, 2019 Available: https://www.csoonline.com/article/3387146/what-is-emotet-and-how-to-guard-against-this-persistent-trojan-malware.html

[6] R. Çelik, A. Gezer "Behavioral Analysis of Tricot Banking Trojan with its New Tricks" International Journal of Technology and Engineering Studies.Available:https://kkgpublications.com/wpcontent/uploads/2019/12/ijtes.5.10004-3.pdf

[7] Alexander S. Gillis, K. Elissa "TrickBot Malware, After Emotet takedown, TrickBot roars up threat charts" Technical Writer and Editor in ComputerWeekly Available:https://www.techtarget.com/searchsecurity/definition/TrickBot-malware

[8] David Garcia "Vadokrist: Banking Malware Targeting Brazilian Entities" Fer. 17, 2020 Available:https://www.revelock.com/en/blog/vadokrist-banking-malware-targeting-brazilian-entities

[9] Revti Vadjikar "Top 4 Ways Emotet Breaches Banking Security" Factspan, January 15, 2018 Available: https://www.factspan.com/top-4-ways-emotet-breaches-banking-security

[10] PCrisk Team "Emotet Blunders through Attack Campaign" PCrisk, 24 September 2020 Available: https://www.pcrisk.com/internet-threat-news/18929-emotet-blunders-through-attack-campaign

[11] A Gezer, G Warner, C Wilson, P Shrestha "A flow-based approach for TrickBot banking trojan detection" Computers & Security, 2019 Elsevier Van Bladel, *Electromagnetic Fields*, John Wiley & Sons, 2007, p.1176. Available:https://aperta.ulakbim.gov.tr/record/111954#.YZXwNWBByUk

[12] Aditya K. Sood, Richard Enbody "Multi-staged Attacks Driven by Exploits and Malware" in Targeted Cyber Attacks Malware Infection, April18,2014 Available:https://www.elsevier.com/books/targeted-cyber-attacks/sood/978-0-12-800604-7

[13] Malware Analysis by Hasherezade on 29 Dec 2017

Available:https://github.com/hasherezade/malware_analysis/tree/master/trickbot
[14] Frederick Lardinois "Google Wants to Speed Up the Web With Its Quic Protocol" Techcrunch, 3 April 18, 2015 Available: https://www.ajer.org/papers/v6(04)/F06044045.pdf
[15] Steve Patrick "Network protocols, What are QUIC? Everything You Need to Know" in APNIC, September 14, 2021 Available: https://blog.apnic.net/2019/03/04/a-quick-look-at-quic/
[16] Leslie F. Sikos "Forensic Science International: Digital Investigation" Volume 32, March 2020, 200892 Available:https://www.sciencedirect.com/journal/forensic-science-international-digital-investigation/vol/32/suppl/C
[17] Sunghoon Lee "Using Weka in Matlab" version 1.5 Mathworks Jul 22, 2015 Available:https://www.mathworks.com/matlabcentral/fileexchange/50120-using-weka-in-matlab
[18] Nir Shwarts, Kessem L. "Trojan Widens Its Attack Scope in Spain, Brings Redirection Attacks to Local Banks" Security Intelligence, July 19, 2017 Available:https://www.imperva.com/learn/application-security/dns-hijacking-redirection/
[19] Katsumi Ono, Isamu Kawaishi, Toshihiko Kamon "Trend of Botnet Activities" Proceedings of the 41st Annual IEEE International Carnahan Conference on Security Technology, Canada (2007), pp. 243-249, November 2007 Available: https://ieeexplore.ieee.org/document/4373496
[20] J. Davison "TrickBot Banking Trojan Adapts With The New Module" Webroot Threat Lab, March 21, 2018 Available: https://www.webroot.com/blog/2018/03/21/trickbot-banking-trojan-adapts-new-module/
[21] Marc Salinas, Jose Miguel Holguin "Innovation in Process Malware Report Evolution of TrickBot" June, 2017 Available:https://www.slideshare.net/rootedcon/jose-miguel-holguin-marc-salinas-taller-de-anlisis-de-memoria-ram-en-sistemas-windows-rooted2019
[22] Liu, J., Xiao, Y., Ghaboosi, K., Deng, H., Zhang, J. "Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures" EURASIP Journal on Wireless Communications and Networking. Volume 2009. Available: https://doi.org/10.1155/2009/69265
[23] Yusuf Sönmez, Meltem Salman and Murat Dener "Performance Analysis of Machine Learning Algorithms for Malware Detection by Using CICMalDroid2020 Dataset" Available:https://dergipark.org.tr/tr/download/article-file/2060165

BIOGRAPHIES

RÜVEYDA ÇELİK was born in Kayseri City, Turkey, in 1996. Sreceived bachelor degree in Electrical and Electronics Engineering from Niğde Ömer Halisdemir University Niğde, TURKEY, in 2019. She started master's degree in Electrical and Electronics Engineering at Kayseri University in 2019 and still continues.

Her research interests include internet traffic analysis, network traffic modelling and characterization, malicious banking viruses, data mining, telecommunication technologies, IoT systems, and malware analysis.

ALİ GEZER was born in Kayseri City, Turkey, in 1976. He received the B.S. degree in Electronic and Computer Education from Marmara University in 1999 and M.S. degree in Computer Engineering from Erciyes University in 2004, and the Ph.D. degree in Electronic Engineering from Erciyes University, Kayseri, TURKEY, in 2011. He is an Associated Professor with the Electronic and Communication Technology in Kayseri University.

His research interests include internet traffic analysis, self-similarity, network traffic modelling and characterization, signal processing techniques, telecommunication technologies, IoT botnet investigations, and malware analysis.