

RUSTSCAN

Table of Contents:

Introduction	2
Tool Details	2
GitHub Repository/Source	2
Dependencies	2
Use Cases List	2
Version of the Tool	2
Operating System Used.....	2
Interface	3
Additional Details.....	3
3. Installation of the Tool	3
4. Execution.....	4
RustScan Commands and Options.....	4
Basic Command	4
Common Options.....	4
5. Output.....	5
• If RustScan is installed:	5
• If RustScan is not installed:	5
• If running on Linux and installation succeeds:	5
• If running on Linux and installation fails:	5
• If not running on Linux:	6
• If RustScan runs successfully:.....	6
• If the command times out:	6
• If an error occurs while running RustScan:	6
• Prompt for user input:	6
• If an error occurs while running RustScan:	6
6. Automation	7
7. DeepDive Research.....	7
Use Cases List:	7
Performance and Efficiency	7
Key Features of RustScan.....	8
8. Conclusion	8
Key Takeaways:	8
Community and Development	8

Introduction

RustScan is a high-performance port scanner developed in the Rust programming language. It is designed to provide extremely fast and efficient scanning of network ports, surpassing the speed of traditional port scanners like Nmap while maintaining low resource usage. RustScan combines the speed of Rust with efficient multi-threading and seamless integration with Nmap, making it a powerful tool for network security professionals.

Tool Details

GitHub Repository/Source

The source code for RustScan is available on GitHub: [GitHub Repository: RustScan](#)

Dependencies

- **Rust:** The programming language used to develop RustScan. You need Rust installed to build from source.
- **Nmap:** RustScan can use Nmap for deeper scans after initial port scanning.

Use Cases List

1. **Network Security Audits:** Quickly scan large networks to identify open ports.
2. **Vulnerability Assessment:** Combine RustScan's speed with Nmap's detailed scanning capabilities to assess vulnerabilities.
3. **Penetration Testing:** Use RustScan in penetration testing workflows to quickly identify potential entry points.
4. **System Administration:** Monitor and manage open ports on critical infrastructure.
5. **Compliance Checking:** Ensure compliance with security policies by regularly scanning for open ports.

Version of the Tool

This report covers RustScan version 2.2.3.

Operating System Used

- Kali Linux

Interface

- **CLI (Command-Line Interface):** RustScan operates entirely through the command line, providing powerful scripting and automation capabilities.

Additional Details

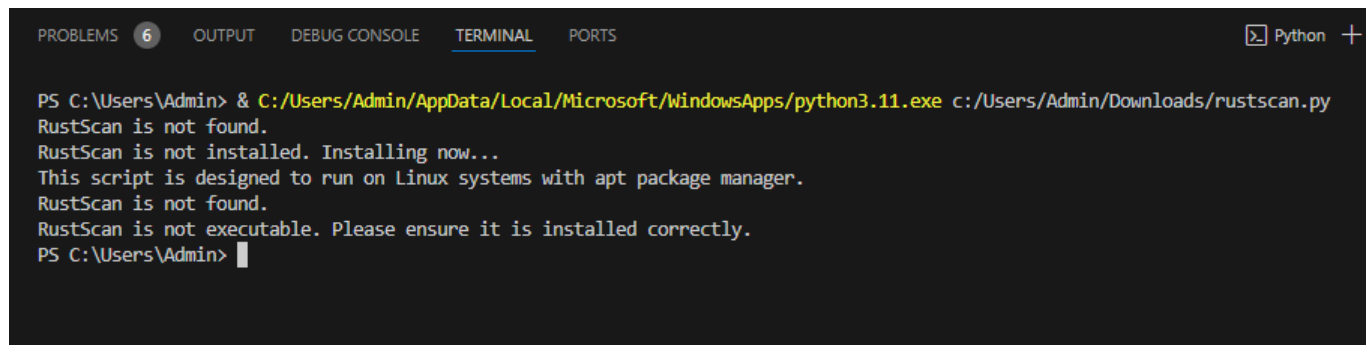
- **Speed:** RustScan is known for its extremely fast scanning capabilities, often scanning all 65,535 ports in a matter of seconds.
- **Integration with Nmap:** RustScan can pass open ports directly to Nmap for further analysis, combining speed with detailed scanning.
- **Ulimit Management:** RustScan provides options to manage file descriptor limits, ensuring optimal performance even on systems with restrictive limits.

3. Installation of the Tool

The tool that I've designed doesn't require any kind of permission but the os should be Kali Linux or any other linux distro for the script to work.

As soon as the script runs it first checks for the operating system its executed on and if its an unsupported version, it will throw an error and automatically exit.

Here's an example-



```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python +
PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Admin/Downloads/rustscan.py
RustScan is not found.
RustScan is not installed. Installing now...
This script is designed to run on Linux systems with apt package manager.
RustScan is not found.
RustScan is not executable. Please ensure it is installed correctly.
PS C:\Users\Admin> 
```

And if the script is run in an a linux environment then it will check whether rustscan is installed, and if not then it will automatically run

‘Sudo apt install rustscan -y’
and install rustscan on the machine.

4. Execution

To use RustScan with the provided Python script, here is a detailed mapping of the program and tool commands:

1. Environment Preparation
Since the installation of rustscan is already done in the previous step, the environment is already prepared for execution.
2. User Input
The user is then prompted for the input of the target ip address or the target web address.eg- 192.168.0.1,www.google.com
3. Execution of code
Then the code is executed and runs rustscan in the terminal and gives the output

All of these steps are better explained and visualized in the attached video

RustScan Commands and Options

Basic Command

- Scan a single IP address:

```
rustscan -a 127.0.0.1
```

Common Options

- Specify the target IP address or range:

```
rustscan -a 192.168.1.1
```

- Specify the ports to scan:

```
rustscan -a 192.168.1.1 -p 22,80,443
```

- Scan all ports:

```
rustscan -a 192.168.1.1 -p 1-65535
```

- Specify a range of ports to scan:

```
rustscan -a 192.168.1.1 -p 1-1000
```

Advanced Options

- Set the number of retries for scanning:

```
rustscan -a 192.168.1.1 --retries 3
```

- Set the timeout in milliseconds:

```
rustscan -a 192.168.1.1 --timeout 1000
```

- Specify the rate (number of packets per second):

```
rustscan -a 192.168.1.1 -r 1000
```

- Use a specific scanning script:

```
rustscan -a 192.168.1.1 --script /path/to/script
```

- Integrate with Nmap for service detection and version detection:

```
rustscan -a 192.168.1.1 -- -sV
```

- Specify the output format (e.g., JSON):

```
rustscan -a 192.168.1.1 --out JSON
```

- Specify the file to save the output:

```
rustscan -a 192.168.1.1 --out-file results.json
```

5. Output

- If RustScan is installed:

RustScan found at: /usr/bin/rustscan

- If RustScan is not installed:

RustScan is not found.

- If running on Linux and installation succeeds:

RustScan installed successfully.

Output: (installation output details)

Error: (installation error details if any)

- If running on Linux and installation fails:

Error occurred while installing RustScan: (error details)

Output: (captured output)

Error: (captured error)

- If not running on Linux:

This script is designed to run on Linux systems with apt package manager.

- If RustScan runs successfully:

Running command: rustscan -a target_ip -r port_range -b batch_size --ulimit ulimit -t timeout -- nmap_options script_options

(RustScan standard output)

(RustScan standard error)

Command executed: rustscan -a target_ip -r port_range -b batch_size --ulimit ulimit -t timeout -- nmap_options script_options

Output saved to: target_ip.txt

- If the command times out:

Error: Command 'rustscan -a target_ip' timed out.

- If an error occurs while running RustScan:

Error occurred while running RustScan: (error details)

(RustScan standard output)

(RustScan standard error)

Output saved to: target_ip.txt

- Prompt for user input:

Please enter the target IP address:

Please enter the port range (default 1-65535):

Please enter the batch size (default 1500):

Please enter the ulimit value (default 5000):

Please enter the timeout in milliseconds (default 2000):

Please enter any additional Nmap options (e.g., -sC -sV):

Please enter any additional script options (e.g., -g --ignore-xml-errors):

- If an error occurs while running RustScan:

Error occurred while running RustScan: (error details)

(RustScan standard output)

(RustScan standard error)

Output saved to: target_ip.txt

6. Automation

This script gives output for all the important and relevant commands that rustscan can give to. Also after execution of this script the output of the tool is displayed on the terminal and also stored as a .txt file with the name based on the ip address or web address entered. Eg- “192.166.10.15.txt”

7. DeepDive Research

RustScan is a highly efficient and fast port scanning tool designed to work alongside Nmap. It significantly reduces the time required to find open ports by quickly scanning them and then passing the results to Nmap for a detailed scan. RustScan is developed to bridge the gap between discovering open ports and performing in-depth scans, enhancing the overall port scanning process.

Use Cases List:

- Fast initial port scanning
- Integration with Nmap for detailed service and version detection
- Capture the Flag (CTF) competitions
- Network reconnaissance during penetration testing
- **Network Audits:** Quickly scan large networks to identify open ports and services.
- **Vulnerability Assessment:** Integrate with Nmap to perform detailed vulnerability scans.
- **Routine Security Checks:** Regularly scan critical infrastructure to ensure no unexpected services are running.
- **Penetration Testing:** Use RustScan for the initial reconnaissance phase to gather information about potential targets.

Performance and Efficiency

RustScan's performance is one of its standout features. It achieves high-speed scanning by:

- Using asynchronous I/O to handle multiple scans simultaneously.
- Reducing overhead through efficient memory and CPU usage.
- Implementing smart algorithms to adapt scanning behavior based on network conditions.

Key Features of RustScan

1. **Speed:** RustScan is designed to be extremely fast, capable of scanning all 65,535 ports in just a few seconds.
2. **Efficiency:** Uses intelligent adaptive scanning techniques to minimize unnecessary resource usage.
3. **Integration with Nmap:** Allows for deeper analysis and detailed reporting by using Nmap's capabilities after initial scanning.
4. **Easy to Use:** Simple command-line interface with straightforward options.
5. **Flexibility:** Offers a variety of scanning options and configurations.

RustScan's unique capability of quickly identifying open ports and passing them to Nmap for detailed analysis makes it an invaluable tool for network scanning and penetration testing

8. Conclusion

RustScan is a powerful tool that revolutionizes the way port scanning is performed by significantly reducing the time required to identify open ports and integrating seamlessly with Nmap for detailed analysis. Its ability to quickly scan all 65,535 ports and pass the results to Nmap makes it a valuable asset for cybersecurity professionals, penetration testers, and network administrators.

Key Takeaways:

- **Efficiency:** RustScan's speed is its primary advantage, allowing for rapid identification of open ports.
- **Integration:** By working in conjunction with Nmap, RustScan offers detailed insights into services running on open ports.
- **User-Friendly:** The tool's CLI interface, along with flexible configuration options, makes it accessible for users with various levels of expertise.
- **Open-Source:** As an open-source tool, RustScan benefits from community contributions and continuous improvements.

Community and Development

RustScan is an open-source project with an active community. Contributions are welcome, and the development is ongoing to add more features and improve existing functionalities. You can find the source code, contribute, or report issues on the [RustScan GitHub repository](#).

9. References

- [GitHub](#)
- [PenTest Tools](#)
- [RustScan GitHub Repository](#)

