Introduction and Purpose

This project focused on creating a custom tool that allows for automated and simplified use of network scanning tools. Our final tool has been simply titled the “Automated Network Scanner”, or “ANS” for short, to highlight the easily understandable nature of the tool for users who are unfamiliar with the concepts. The ANS is a Bash-based tool that creates a guided experience for network scanning reconnaissance. It utilizes functionality from nmap, masscan, and netcat for a rounded and thorough compilation of scan results from common network scanning utilities in the field. Scan results from the ANS tool are saved to a desired file output type that allows for easy review of results and serves as a backup of the scan results.

The overall purpose of this tool is to create an easier way to conduct network scans and analysis while preserving the functionality of the original utilities. For many that are new to the IT field, using new tools can be a daunting task and it can be difficult to start learning how to properly utilize them. Our tool aims to solve this issue by guiding the user through the setup of each tool and their specific options with a text-based user interface. The user can choose common setups for each utility or choose to customize the desired flags. Once the user has fully configured the tool, it will automatically run through a scan on each tool with the selected settings and output the results to the specified output file type(s). These output files can be either a CSV, TXT, or HTML file, as well as any combination of those files. These output files (CSV and HTML specifically) highlight certain services/ports based on the severity of exposing a certain service, enabling users to quickly see which scan findings are the most important, as well as learn what to look for in the future. While the majority of functionality is intended for new users, individuals with more advanced knowledge can also use this tool manually to automate the scans all at once without having to run multiple scan types independently.

Technical Implementation

The ANS is built using Bash scripting and organizes configuration input, utility execution, scan data analysis, and data report generation. This implementation can be described in terms of its architecture, dependencies, interaction modes, scanning processes, and reporting workflow.

Overall Architecture

The ANS script functions as a multistage pipeline consisting of the following components: Configuration and Input, Tool Selection and Flag Management, Scanning Engine, Temporary Storage, Analysis and Alert Engine, and the Reporting and Packaging System. Each of the parts are implemented using dedicated functions, which allows for code readability and makes it easier to add new features. The first part, configuration and input, is in charge of collecting the user's choices before scanning begins. It has two different modes of user interaction, depending on the user's experience level. 'Guided Mode' is an interactive mode that prompts the user for input to configure the tool step by step. This mode is ideal for users with beginner level knowledge. The 'Advanced Mode' allows users to input all configurations and flags in one line using pipe characters to separate each section. The scanner then parses the segments and applies them to the corresponding tools/configuration sections. This mode is recommended for users who have advanced knowledge of the utilities but still seek a form of automation. Once the user has finished IP configuration, the tool selection and flag management stage begins. The user can choose nmap, masscan, netcat, or any combination of the tools, as well as selecting presets for each tool or using custom flags. A configuration summary is always displayed before scanning so that the user may double check their settings without having to rerun the scan. The scanning engine then runs each selected utility using the user-configured flags and settings. Raw output from the scans is stored in separate files under '/tmp' within a temporary dedicated directory. Error handling makes sure that even if one utility has a problem, the script continues running the remaining utilities until completion. Next, the temporary storage layer isolates the intermediate output before it is processed. The results from each tool are stored in files with the naming convention 'utilityname_results.txt'. Once the script has reached completion or been interrupted, the temporary files are cleaned up. The analysis and alert engine checks the scan results for common security risks/patterns and marks any that pose a potential security risk. Some of these marked risks are things like open SSH ports, exposed web services, database ports, and any unusually high number of open ports on a target. When these patterns are detected, the script prints an alert to the terminal and logs it in an alert file. Finally, the reporting and packaging system generates the outputs that were selected previously by the user. These outputs range from TXT, CSV, HTML, or any combination of the three types. These outputs are then stored in a timestamped directory, which is then zipped to conserve space. The original unzipped directory is deleted to conserve space further. In addition to the output files, an 'alerts.log' file is stored in the zipped directory for a simple list of security risk alerts that were thrown during the scanning process.

Tools and Dependencies

The ANS Bash script relies on a set of external tools and utilities that are standard for most Linux distributions. The required external tools are as follows: nmap, masscan, netcat, sudo, and zip. The utilities that are standard for Linux are as follows: awk, bash, cat, cd, date, echo, grep, mkdir, printf, rm, sed, timeout, tr, xargs. If any dependencies are not available when the tool is being configured, the tool will throw an error with the name of the package that must be installed for proper functionality.

Justification and Analysis

The design of the Automated Network Scanner addresses both educational and practical needs. Students learning cybersecurity must be able to perform and understand network scanning, result interpretation, and risk identification. However, traditional scanning tools can be intimidating due to their complexity and amount of unstructured data they produce. One of the main motivations behind the project was to lower the barrier of entry for new students that want to use network scanning tools. The guided mode walks the user through every configuration decision and helps them to learn which tools perform which functions and the purpose of certain flags for different scans. Meanwhile, advanced mode serves the needs of an experienced user by allowing full customization and automation without unnecessary prompts. The ANS script uses a modular architecture because it allows beginners to understand the workflow of configuring and use the tools step by step. Each function is responsible for exactly one part of the process, which reflects good scripting techniques taught in the IT field. Using Bash as the scripting language was a justified decision because it is native to Linux, which is very common in cybersecurity, as well as being ideal for automating other command line tools. This aligns with real world IT practices where shell scripts are used to wrap common utilities into automated workflows. The decision to store results in a timestamped folder that is automatically zipped ensures that each scan is easy to archive and share. This organization is especially valuable in settings where an individual submits scans or needs to compare results, like a student in a classroom. The inclusion of the ‘alerts.log’ file provides a simplified interpretation of the results that highlights risks that could pose critical security threats in real-world situations. This helps students understand not only what ports are open, but why specific ports may be more sensitive or dangerous than others. Out of all the output file types, we put the most thought into the HTML design because of its visual capabilities. The HTML report was designed to be visually readable and beginner friendly. It provides color coded tables, summaries of high-risk findings, and collapsible sections containing the raw output of each scan type selected. This was done to improve comprehension of scan information, as well as helping users understand the transition from raw data to something that is like a structured report. Although the script is effective for its basic purpose, there are some limitations. For example, the masscan utility often requires root privileges, which causes the user to have to input their password between scanning utilities. Some larger scans may also require more space when producing a large HTML report file. Additionally, the risk analysis is based primarily on port numbers and does not incorporate deeper vulnerability information. This could be addressed by the addition of a vulnerability scanning utility, but we refrained from adding this as we felt it exceeded the scope of our original project idea. These limitations are acceptable for the purpose of our educational project but could be addressed in future revisions. From an ethical perspective, our script emphasizes the need for authorized scanning and responsible use. Users are reminded that scanning machines without permission is illegal and inappropriate in both academic and professional settings. This reminds users of the importance of good judgement and compliance with legal and ethical guidelines. Overall, the architecture, design decisions, and educational focus of the project are justified by the need for a user-friendly and highly organized method of performing network reconnaissance.

Conclusion

The Automated Network Scanner successfully met its goals of accessibility, clarity, and educational value. It provides an intuitive way for beginners to explore network scanning tools while still supporting advanced customization for advanced users. By integrating nmap, masscan, and netcat into a single automated pipeline, the script streamlines common reconnaissance tasks and produces a structure, multi-format report that is easy to interpret and archive. From a technical perspective, the project demonstrates a strong command of Bash scripting, modular function design, tool integration, parsing, reporting, and file management. From an educational perspective, it models ethical scanning practices, improves comprehension through organized reporting, and supports both guided learning and independent exploration. While there are opportunities for expansion, the current version of the scanner already represents a polished, comprehensive, and highly practical tool for its use case. If desired, future revisions can incorporate more robust risk scoring or tighter integration with system logging and monitoring frameworks. However, the Automated Network Scanner fulfills its core mission: making network reconnaissance understandable, consistent, and accessible for learners and seasoned professionals alike.