Weapon X – Automated Reconnaissance & Enumeration Framework

Final Project – Written Report

IT 359, Section 001

December 4, 2025

Introduction

Our group, the "Legendary Beasts," decided to figure out a faster solution to scanning a full network as well as the very beginning stages of pen testing with automation, and after producing this very basic idea, we decided to develop the tool "Weapon X." The problem we wanted to solve, and the reason we decided to set out on this idea, was because we were at the very beginning stages of our Hack The Box work and were quickly getting tired of running Nmap scans, then having to parse the results for what tools we needed to use, and then having to consult our playbook on what to do and how to do it.

The idea of developing it to work across networks came from Kevin after he discovered the tool "ScanCannon" from a security researcher, JohnnyXmas, through the group Burbsec, which is a collective of different security professionals ranging from students like us to C-level executives from the Chicago suburbs who are dedicated to cybersecurity. Another part of the inspiration for developing this tool is that we are up-and-coming professionals in the space, so having a tool like this could not only be helpful for any potential red teaming work we might pick up down the road, but also could be something that looks phenomenal on a resume because it demonstrates knowledge of programming and foundational security concepts. It can also be used to help people who are just starting out in the space, because the tool we developed is very

verbose when it comes to output, so it can actually show the steps necessary to start pen testing, which is always the hardest part of this field.

Technical Implementation ("The How")

For our code, we decided to go with Python given the experience that we already possess in the language as well as the ease of calling different terminal commands from the program, which is what a portion of our code does. The main portion of our code, however, is parsing through the results of ScanCannon. Using Python's standard library, we were able to extract the text that we needed, which was the IP, port, and the name of the service, to help with understanding what was going on for people that might not have every port number on Earth memorized.

For the parsing part of what we did, we used built-in string-handling commands to strip the text file down to the bare bones and allow us to break the IPs and ports apart before sending them to the dataclasses library to help decide what should be done next. When it came to navigating the directory, we decided that the best route was to use the pathlib library because it was the best one across the board according to Python docs and some Reddit forums that we read.

We also added a custom configuration file written using JSON, which allows the user to fully customize things like tools used or wordlists for people who might already have those in an enterprise environment. The fun part starts after it has gone through the results: it stores the IPs and ports, and then looks at how the user has set up the port-service map to help normalize input to the command-line prompts. It then takes whatever valid ports and IPs it finds and runs the

commands based on the current IP, selecting the service according to the port that it is on. This just incrementally works through the ports list as part of the automation process.

One big roadblock that we were unable to get around during the project was that Metasploit has a remote procedure call tool, which we wanted to use to expand the toolset of what we were doing. However, the documentation surrounding the RPC tool was underwhelming and provided no clarity as to what we were doing. Calling Metasploit normally from the console was also giving us problems, and because loading Metasploit takes time (which the computer does not know), it kept producing funky results and commands. So, we decided to just use every other tool that we had experience with besides Metasploit.

Another downside is that we were unable to pipe a direct call to ScanCannon into the program, which meant that the user has to have ScanCannon installed separately, run it, and then put the output into the parser manually, which takes time. Being able to automatically pipe it into the workflow would have made the whole process go faster, but that's something that we would've liked to implement in the future. Something else that we would have liked to see done is a more simplistic approach to choosing wordlists so that you don't have to edit the configuration file every time you want to test a new wordlist and could instead just tack it onto the actual execution of the program like every other Python or security tool.

Justification & Analysis ("The Why")

When our group first started doing Hack The Box work, we quickly realized how much time we were wasting on the same repetitive early-stage tasks. Every box started the exact same way: run an Nmap scan, wait, sort through the results, figure out what tools we need, and then check our playbook to remember the steps. The work that we were doing wasn't hard by any

stretch of the imagination. It's just that to us the work felt boring and unexciting. It especially felt tedious at certain points when it came to doing the actual work when we had to switch between VMs, writeups we were doing and notes we were reading about the tools. The process as a whole felt boring and repetitive and something that could easily be automated. In real pen testing or security operations, that early recon stage needs to be quick and efficient. But as students trying to learn the fundamentals, we were spending way too much time on the low-level setup instead of the interesting analysis that comes after.

The idea for Weapon X really started when Kevin came across a tool called ScanCannon, made by security researcher JohnnyXmas and shared through the Burbsec community. Seeing professionals automate the exact same recon steps we were struggling with made something click for us: these frustrations were not just “beginner problems.” Even experienced pen testers automate the boring parts, so they can focus more on the heavy lifting.

So, we set out to create Weapon X to streamline and speed up the easy but tedious reconnaissance and enumeration process. The goal is simple: speed up scanning, automatically organize results, and even automatically try out the low hanging fruit exploits based upon the results of the scan. By automating the tedious and repetitive tasks, we can reduce wasted time and help newer users get comfortable and gain some confidence with the flow of pen testing without getting stuck on the basics.

In short, Weapon X exists because the beginning of a pen test does not need to be painful. We wanted a way to get from “start scanning” to “I know what to do next” as fast as possible, and since no beginner-friendly tool like that really existed, we decided to build it ourselves.

Conclusion

In the end, Weapon X was a solid class project that pushed us to think differently about the early stages of pen testing. It helped us see how much time can be saved by automating basic recon tasks, and it gave us a chance to apply what we have learned in IT 359 in a hands-on way. Building the tool forced us to break down our usual workflow, understand where the slowdowns were, and design something that could make that process smoother.

Even though Weapon X is not meant to be a fully professional tool, working on it helped us get more comfortable with Python, calling system commands, handling scan output, and thinking through how a pen testing workflow actually operates behind the scenes. Another massive positive for the project was that it gave us more exposure to tools that we hadn't worked with in class like Nitko, to help with web app scanning, or WhatWeb, to aid in discovering what technologies are under the hood of different websites. This project gave us the experience of collaboration as a group, dividing tasks, as well as fundamental project development like creating new ideas, testing them and implementing them. Overall, this project gave us the opportunity to create a practical tool for the real world.

To close, we took away a better understanding of enumeration, automation, and the value of streamlining repetitive tasks, skills that will be useful as we continue learning and developing our cybersecurity careers as well as fundamental project management that will also enable us to grow professionally.