William Walker
Professor Fuqua
CDT
February 15th, 2026

Information Gathering Red Team Tool - WBW1991 Report

Section 1: Tool Development

Q1: What tool did you create and what category does it fit into? Explain your choice and why this tool fills a gap in your Red Team capabilities.

I created a tool that consolidates and automates reconnaissance and information gathering on a compromised Linux host. Instead of manually running dozens of commands, checking multiple directories, and digging through logs or config files, my tool pulls from all the known places where secrets, credentials, misconfigurations, and operational clues tend to hide, putting everything into one unified JSON file.The idea is to give a Red Team user a complete snapshot of the machine in seconds.

This tool fits Category 5: Information Gathering / Credential & Secret Collection. It does not modify the box or exploit it.

FullInfoScan is built from multiple modules, each responsible for a specific area:System metadata, SSH keys and configs , Recon (services, ports, packages), Process inspection , Log harvesting , Environment variables, Config files ,Bash history. All of these feed into a single JSON output. That JSON is extremely powerful, it contains almost everything you'd want to know about a compromised Linux box, and it can be moved, stored, or analyzed later.

The tool also includes JSONScan.py, which takes the FullInfoScan output and filters it down to the most likely useful or sensitive findings. It acts like an automated grep engine for secrets, tokens, credentials, and suspicious strings. This makes the tool useful both for quick recon and deeper post-collection analysis.

Overall, this tool fills a gap in my Red Team capabilities because it gives me a fast, modular, repeatable way to gather everything I need without manually running commands or missing important artifacts.

Read README.md for more in depth info!

Q2: Describe the technical implementation. What language did you use and why? What were the major technical challenges you encountered and how did you solve them?

I wrote the tool in Python, mainly because I'm very familiar with it as a computer science major and because Python is perfect for scripting, file parsing, and running system commands. It also makes modular design extremely easy, which was important for keeping the tool clean and

William Walker
Professor Fuqua
CDT
February 15th, 2026

maintainable. The biggest technical challenges were figuring out which commands and files were the most valuable to target, organizing the output so it wasn't a giant mess, and making sure everything stayed modular. I had to research what commands give the most useful recon data and where Linux stores important artifacts like SSH keys, logs, and configs. Structuring everything into readable JSON was also important so the output wasn't overwhelming. Another challenge was handling unreadable files gracefully, some files require root, so instead of crashing, the tool marks them as "unreadable" in the JSON. Overall, I'm thankful I chose Python as it made the whole process smooth, and the modular design kept the tool easy to test and expand.

Q3: How does your tool provide operational security advantages over existing tools? What makes it stealthy or hard to detect?

I don't know many existing tools in general, but I think mine fills a useful niche. The main advantage is that it gives you one single file that contains almost everything you could want from a compromised Linux box. You don't need to stay on the machine long, you can run the tool, grab the JSON, and leave. It's fast, disposable, and easy to clean up. You can delete the directory afterward or move the JSON somewhere else, and nothing needs to remain on the target.

In terms of stealth, the tool doesn't modify anything on the system. It only reads files and runs standard Linux commands. It doesn't create logs or network traffic. The interesting part is that it runs a lot of commands very quickly, which creates a noticeable burst of activity. While that might seem like a downside, it actually works in Red Team's favor: the sheer volume of harmless system queries can overwhelm logs and make it harder for defenders to pinpoint what the user was doing. So even though it's not silent, it's extremely fast and creates enough noise to make analysis harder.

Q4: What testing did you perform to verify your tool works correctly? Describe your test environment and test cases.

To test the tool, I spun up a default Ubuntu VM in OpenStack and SSHed into it through VS Code. I tested each module individually as I built it, saving example outputs in an examples/ directory so I could compare results as I refined the tool. It was easy to tell whether something worked, I just checked the JSON to see if the expected data appeared. If a command didn't return what I wanted or a file wasn't being read correctly, it was immediately obvious.

My testing process was simple but effective i would say. I ran each module, verified the JSON, fixed anything that looked wrong, and repeated. I also tested how the tool handled unreadable files and confirmed that it reported them correctly instead of crashing. After finishing development, I deployed a brand new VM and ran the full tool from scratch to make sure nothing

William Walker
Professor Fuqua
CDT
February 15th, 2026

was hard-coded or dependent on my development environment. Everything worked as expected, and the JSON output contained all the information that was expected.

Section 2: Team Coordination

Q5: List all team members and what tool each person is developing. Include the team coordination document/spreadsheet.

Team Member	Tool Category	Specific Tool	Programming Language	Target OS	Status
Jean Hughes	Beacon/Callback	Covert Channel (ICMP Based with encryption)	Python	Linux	Assigned
Jack O'Donnell	Service Backdoor	Systemd timer information stash	Python/Bash	Linux	Assigned
Nathan Russell	Destructive/Distracting	Keyboard Layout Randomizer	Powershell	Windows	Complete, possible minor improvements incoming
Koy Monette	Red Team Infra Tool	C2	Powershell/Python	Windows	Assigned
Aidan Morgan	Red Team Infra Tool	Session manager for deploying our tools	Holy C (Something that doesn't need the python interpreter)	Linux	Assigned
William Walker	Information Gathering	Secret and Credential Intelligence Collector	Python	Linux	Assigned
Andrew Xie	Enhancement to Open Source	Impacket (Lateral Movement)	Python	Windows, Linux	Assigned
Seonho Park	Duplicate: Destructive/Distracting	Log flooder	Python	Linux	Assigned

Q6: How did you coordinate with your team to avoid duplication? Describe the process you used to select your specific tool.

Our team coordinated early on to make sure everyone chose a unique tool category and that no two people were building the same thing. We used a shared spreadsheet where each person listed their tool idea, category, language, and target OS. This made it easy to see what areas were already covered and what gaps still existed. During our initial meetings, we talked through each idea and made sure the tools complemented each other instead of overlapping. We also made

William Walker
Professor Fuqua
CDT
February 15th, 2026

sure the tools matched our initial Red Team Roles, mine was Recon lead. Ultimately this helped me settle on an information gathering tool, because nobody else was focusing on Linux recon or credential collection at that level. I also chose it because it fit my strengths and interests, and because I knew it would be useful for the rest of the team's tools.

Q7: How does your tool complement the other tools your teammates are building? Explain how they work together to support Red Team operations.

My tool fits naturally into the rest of the team's workflow because almost every other tool benefits from having solid recon and credential information up front. FullInfoScan gives a complete picture of the compromised Linux host, which can directly support tools like the C2 infrastructure, the session manager, and the lateral movement enhancements. For example, if my tool finds SSH keys, service accounts, or misconfigurations, those can be used by the lateral movement tool or the backdoor tool. If it finds logs or environment variables with credentials, those can feed into the C2 or callback tools. Even the destructive or distracting tools can benefit from knowing what services or processes exist on the machine. Overall, my tool acts as the foundation for the rest of the team's operations by giving them the information they need. It helps everyone work faster and with more confidence because they don't have to manually gather the data themselves.

Section 3: Operational Use

Q8: Provide a detailed scenario of how you would use this tool during a competition.

In a competition setting, once I gain access to a compromised Linux machine, my first priority would be to quickly gather as much information as possible before the Blue Team notices anything. After establishing a foothold, I would transfer the FullInfoScan directory onto the target, usually through SCP, pulling it from our Red Team infrastructure, or just git. As soon as it is on the machine, I would run FullInfoScan.py one time. The tool executes extremely fast, so within a few seconds I would have a complete JSON snapshot of the system. Once the JSON is generated, I would immediately move it off the box, either by exfiltrating it to our Red Team server or copying it to another machine I control. After confirming the file is safe, I would delete the tool directory from the compromised host to reduce artifacts. From there, I can open the JSON locally and begin reviewing the results, or share it with the rest of the team so everyone can look for credentials, SSH keys, misconfigurations, or anything that could help with lateral movement or persistence. If needed, I could also run JSONScan.py on my own machine to quickly highlight the most important findings. The whole process is designed to be fast, simple, and low interaction so I can get in, collect what I need, and get out before defenders catch on.

William Walker
Professor Fuqua
CDT
February 15th, 2026

Q9: What are the risks of using this tool?

This tool does come with some risks, even though it is extremely useful. The biggest risk is that it runs a lot of commands in a short amount of time, which can be noticeable if the Blue Team is actively monitoring logs or process activity. It does not generate network traffic, but it definitely produces a burst of local system activity that could stand out. However, it does not break any competition rules because it only reads files and runs standard Linux commands. It does not modify the system, escalate privileges, or perform anything destructive. If used smartly, it should not leave behind many artifacts besides the JSON file, which can be removed immediately after collection. The main risk is detection, not rule violation, but the speed of execution helps reduce that window.

Q10: What improvements or enhancements would you make given more time?

If I had more time, the biggest improvement I would make is adding regex based secret detection and better filtering to reduce noise. Right now, the JSON output contains a huge amount of information, which is useful, but is very overwhelming. More advanced parsing could automatically highlight only the most important or suspicious items, such as API keys, tokens, passwords, or misconfigurations. This would make the tool even more effective because users wouldn't need to manually sift through as much data. I would also improve the analyzer to categorize findings by severity and confidence level. These features would require more research and testing, especially since I don't have a lot of experience writing complex regex patterns, but they would enhance the tool's usefulness. Even without these improvements, the tool is still valuable because it consolidates everything into one place, but smarter filtering would make it even faster to extract intelligence.

Section 3: Ethical Considerations

Q11: Under what circumstances is it ethical to use this tool? What safeguards prevent misuse? How do you ensure it is only used in authorized contexts?

This tool is only ethical to use in controlled, authorized environments such as competitions, sanctioned Red Team exercises, or training labs where all parties have agreed to the testing. Using it outside of those contexts would be very bad and not cool, because it collects sensitive information from a system without the owner's consent. I didn't build many safeguards into the tool itself, mainly because it only automates actions that a user could already do manually with standard Linux commands. The responsibility comes from how it is distributed and used. I only plan to share it with myself and my Red Team members, and after the competition I will keep the repository private or remove it entirely. Even though the tool is not destructive and doesn't

William Walker
Professor Fuqua
CDT
February 15th, 2026

exploit anything, it still needs to be handled responsibly. The main safeguard is making sure it stays in the hands of people who understand the ethical boundaries and only use it in authorized situations.

Q12: Describe the potential harm if this tool were used maliciously outside of competitions. What is your responsibility as the creator?

If this tool were used maliciously, it could help attackers gather valuable information from real systems much faster and more efficiently. It automates the exact steps someone would take to look for credentials, secrets, and misconfigurations, which could make real world attacks easier. My responsibility as the creator is to limit access to the tool, keep the GitHub repository private after grading, and potentially delete it entirely once the competition is over. I need to make sure it doesn't end up publicly available where someone with bad intentions could use it. Even though the tool isn't inherently harmful on its own, it could absolutely be misused, so I have to treat it carefully and make sure it stays in the right context.