IT 360 Project Report

Bash Script Artifact Tool

By: Ethan Allgier, Jacob Kovar, Owen Reid

Digital Forensics Automation Script Project Report

Introduction

The purpose of our project was to build a simple tool that makes digital forensics work quicker and easier. In many investigations, examiners must run several different programs one at a time to check file integrity, find hidden information, and collect metadata. This can take a long time and increase the chance of making mistakes. Our project solves this problem by combining multiple forensic tools into one automated script. With this script, an investigator only must run one command to get hashes, extract artifacts, and read metadata. This makes the process much faster, more organized, and more consistent.

Technical Implementation

This project was created using bash, which is a scripting language commonly found on Linux forensic systems. Bash is lightweight, easy to use, and works well for automating command-line tools. The script first checks that the proper tools are installed and that the evidence file exists. After that, it creates a special output folder to store all results.

The script uses three different forensic programs:

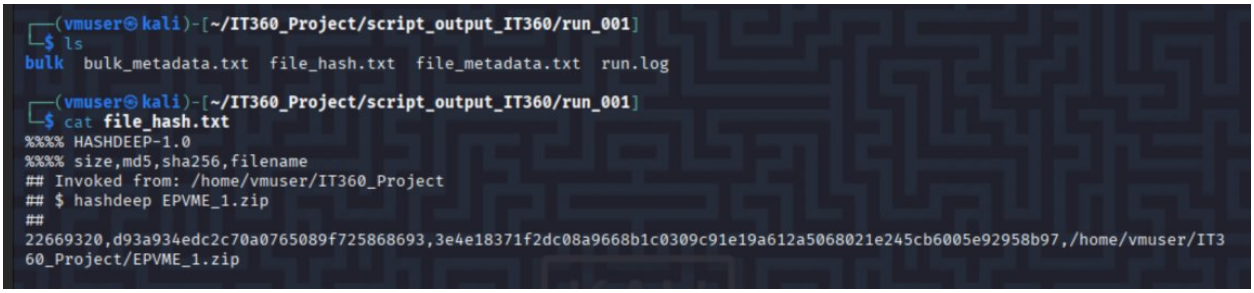
- **hashdeep** – creates MD5 and SHA-256 hash values so investigators can confirm file integrity and see if files have been changed.
- **bulk_extractor** – scans the file for useful artifacts such as email addresses, URLs, domains, credit card numbers, and wordlists.
- **exifTool** – extracts metadata like timestamps, camera information, GPS locations, file types, and author information.

These tools were chosen because they cover different but important parts of forensic analysis: verifying data, finding hidden information, and reading metadata. They also work well together in a single automated workflow.

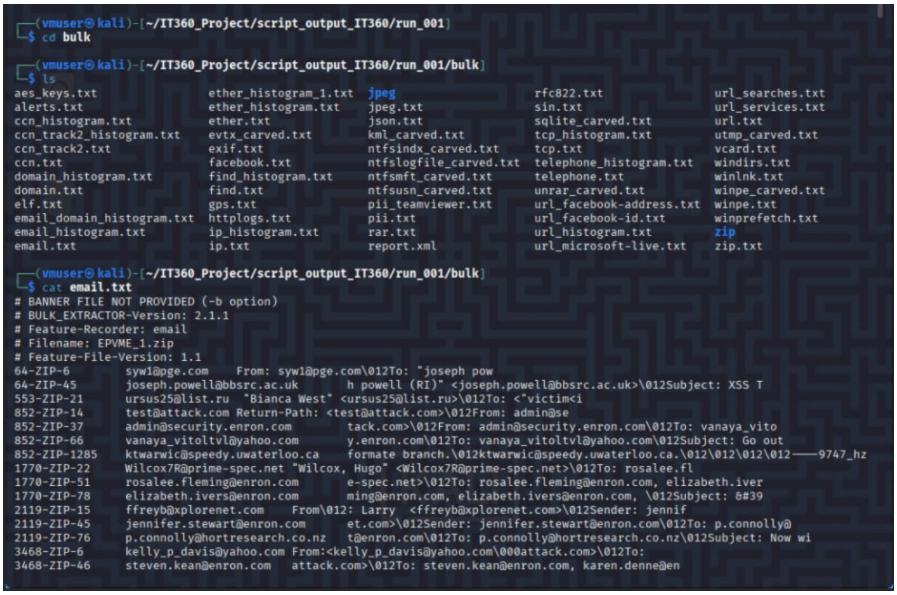
Results

When the script is finished running, it creates a folder called script_output_IT360. Inside this folder you see each run that has happened because the script will figure out the last time the tool has been run. So, when selecting which run you want you will see bulk directory where you see a bunch of different files. All the files that are shown are the default file that bulk can find information based on what you add as your data set. This means some of the files won't have anything in it. So, then we have the bulk metadata file which lets us know which bulk files have information on it. Then you can see the data set was hashed with hashdeep. Then the data set also has a metadata file, letting us get more information on that. Lastly, we have the log.run file just showing what happened when the tool was running.

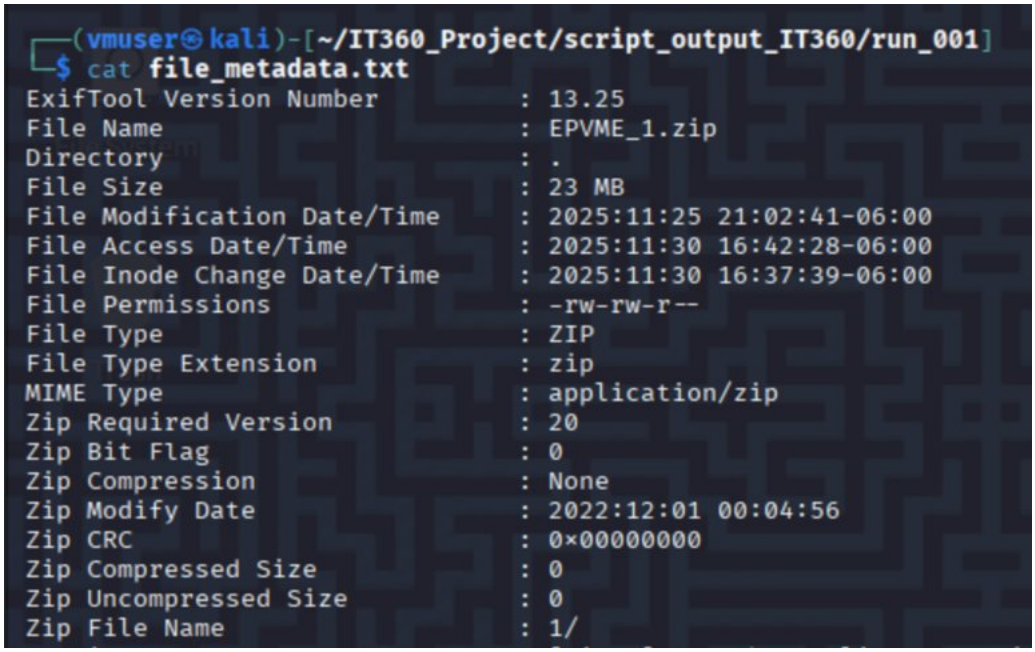
- A text file from hashdeep with the SHA-256 and md5 hash values. *(See the picture below.)*



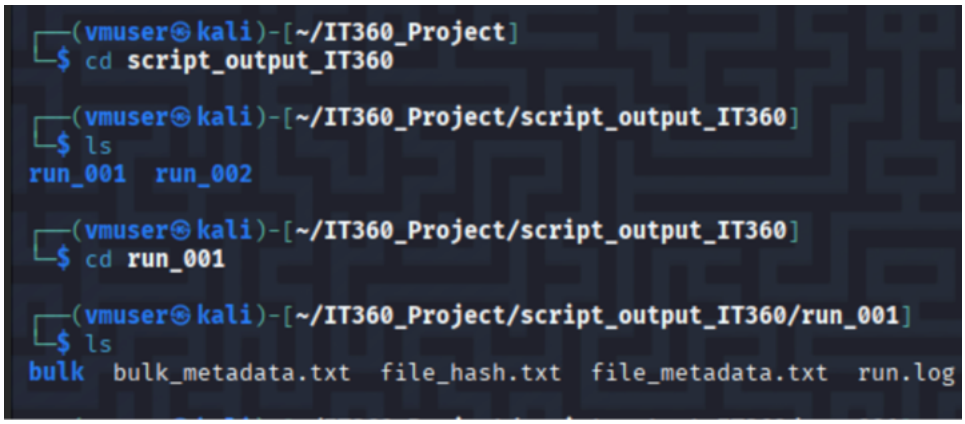
- A bulk_extractor output directory containing files such as email.txt, url.txt, domain.txt, and other lists of artifacts. *(See the picture below.)*



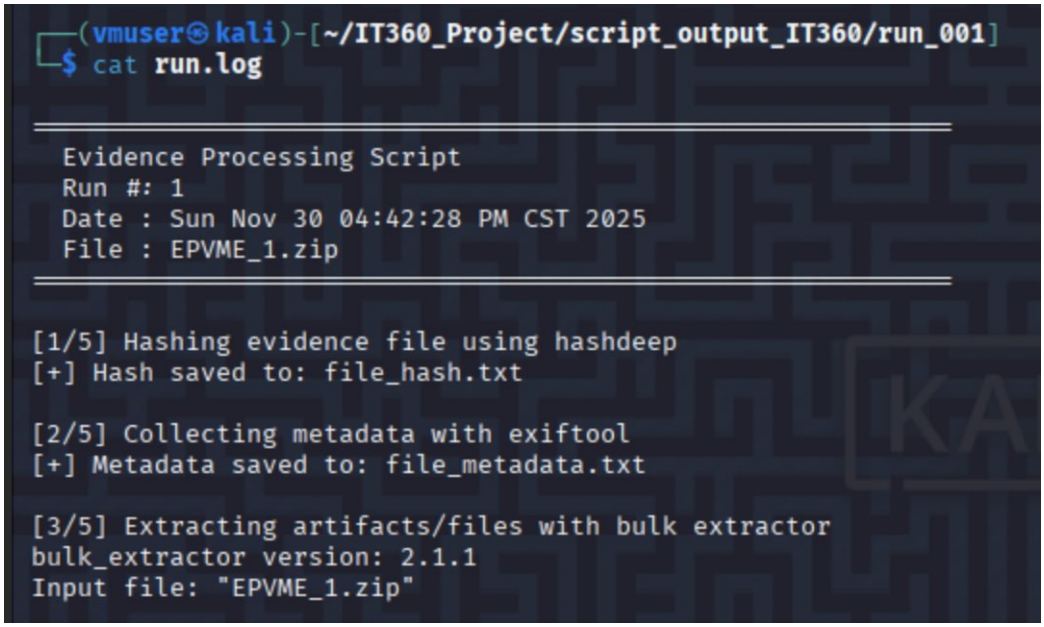
- An exifTool metadata report summarizing file properties and important details. *(See the picture below.)*



- The folder that contains the script output. *(See the picture below.)*



- The run log. *(See the picture below.)*



Conclusion

While working on this project, we learned how valuable automation is in digital forensics. Running one script saved a lot of time compared to running three separate tools. Bash was easy to use for basic automation, but it limited how advanced the interface could be. One challenge we noticed was that bulk extractors sometimes created very large files, which made them hard to sort through without extra filtering. Running the script on large evidence images also took a long time, which showed us the importance of adding progress messages and logging.

One thing we planned at first was to use a tool called ripMIME to extract email attachments from the evidence file. Our goal was to have the script automatically pull out attachments along with emails and other artifacts. However, we could not get ripMIME to work correctly. The attachments did not show up in our tests, even after troubleshooting. Because of this, we decided to remove ripMIME from the project and use exiftool instead. Exiftool worked much more reliably and still gave us useful information. It gives especially metadata like timestamps, file details, and metadata from bulk extractor files to find out exactly which files contain information. In the future, we would like to come back to ripMIME and get it working, because attachment extraction would make the tool even more complete.

If we were to improve this project in the future, we would add features like a summary report, options to choose which tools to run, and stronger error handling. We would also try to include automatic filtering and keyword searching, so investigators do not have to scroll through huge output files. Overall, this project helped us understand how automation can speed up forensic analysis while keeping results organized and reliable.