Logging Intro Excercises

Legend:

Input Command

Output of the previous command

Prerequisites

- Ubuntu 22.04 Server Powered up
- Ubuntu Server on Bridged mode
- From Host OS, ssh to the Ubuntu Server

Fundamental Linux knowledge on Data Streams

In Linux, data streams are channels used for data input and output by programs running on the command line.

There are three primary data streams:

Standard Input (stdin), which is used to feed data into programs.

Standard Output (stdout), where programs output their results or messages.

Standard Error (stderr), specifically reserved for outputting error messages.

By default, stdin reads data from the keyboard, while stdout and stderr display data on the terminal screen.

However, these streams **can be redirected**: you can send the output of a **program to a file**, **pass it as input to another program**, or even **discard it**.

This flexibility allows for powerful and efficient data processing and manipulation in Linux.

In the context of the Linux command line, stdin, stdout, and stderr are three fundamental data streams used by programs to communicate with the outside world, typically the user or other programs.

These streams are a part of the Unix philosophy of using simple, composable tools that perform specific tasks. **Understanding these streams is crucial for effectively utilizing the command line** and scripting in Linux.

*** Note *** Computers start counting at 0! 😀

1. Standard Input (stdin)

- **Identifier**: File descriptor 0 (fd 0).
- **Purpose**: stdin is the standard input stream, which provides data to a program. This stream can be fed from a file, another program's output, or the keyboard input from a user.
- **Use Case Example**: When you use the cat command without specifying a file to read, it reads from stdin, allowing you to input data directly from the keyboard. Typing cat and then entering text followed by Ctrl+D (to signify the end of the input on Unix-like systems) is a direct use of stdin.

2. Standard Output (stdout)

- **Identifier**: File descriptor 1 (fd 1).
- **Purpose**: stdout is the standard output stream, which is used by programs to output data. By default, this data is displayed in the terminal, but it can be redirected to a file or another program.
- **Use Case Example**: In a command like ls, which lists directory contents, the list of files and directories is sent to stdout. This output can be redirected to a file using > operator, like ls > files.txt, or piped into another program with |, such as ls | grep "txt" to filter for files with "txt" in their names.

3. Standard Error (stderr)

- **Identifier**: File descriptor 2 (fd 2).
- **Purpose**: stderr is the standard error stream, which is used by programs to output error messages. By default, error messages are displayed in the terminal, but like stdout, they can be redirected separately. This separation allows users to distinguish between regular output and error messages.
- **Use Case Example**: If you try to cat a non-existent file, the error message (e.g., "cat: file.txt: No such file or directory") is sent to stderr.
    - You can redirect this to a file separately from stdout using 2>, like cat file.txt 2> error.log, or discard it entirely with 2> /dev/null.

Combining and Redirecting Streams

Linux allows the redirection of these streams, enabling powerful combinations and workflows:

- **Redirecting stdout to a file**: command > file
- **Appending stdout to a file**: command >> file
- **Redirecting stderr to a file**: command 2> file
- **Redirecting both stdout and stderr to the same file**: command > file 2>&1
- **Piping stdout of one command to stdin of another**: command1 | command2

Understanding and manipulating these streams are fundamental skills for anyone working with Linux. They enable precise control over the input and output of commands and scripts, facilitating complex data processing, logging, and error handling workflows.

SIEMS and Logs

Computers vs Humans (Skynet approves)

Computers and Security Information and Event Management (SIEM) systems offer several advantages over humans in finding data via queries, particularly in the realms of speed, efficiency, accuracy, and scalability. These advantages are rooted in the inherent capabilities of computer systems and specialized software designed for monitoring, analyzing, and managing security events.

Speed

- **Immediate Processing**: Computers can process vast amounts of data at speeds incomprehensible to humans. They can execute queries and analyze logs spanning billions of records in the time it would take a human to read a single entry.
- **Real-time Monitoring**: SIEM systems can monitor data in real-time, allowing for the immediate detection of anomalies, breaches, or unauthorized activities. This rapid response capability is crucial for minimizing potential damage from security incidents.

Efficiency

- **Automation**: Computers and SIEM systems automate the repetitive and laborious tasks of data analysis. They can run 24/7 without breaks, fatigue, or a decrease in performance. Automation also includes the ability to perform scheduled tasks without human intervention, ensuring that routine checks and analyses are never overlooked.

- **Predefined Criteria**: These systems can be configured with specific criteria for querying and analyzing data, making the search process more efficient. They can filter out irrelevant information and focus on what's important based on predefined parameters.

Accuracy

- **Consistency**: Unlike humans, who may vary in their attention to detail or make mistakes, computers and SIEM systems apply the same level of precision to every task. They consistently follow the defined rules and criteria for data queries and analysis without bias or error.
- **Pattern Recognition**: SIEM systems and machine learning algorithms excel at recognizing patterns and anomalies within large datasets. They can detect subtle, complex patterns that would be nearly impossible for humans to discern, especially in the noise of massive data sets.

Scalability

- **Handling Big Data**: The volume of data generated by modern networks and systems is enormous and continuously growing. Computers and SIEM systems can scale to handle increases in data volume much more effectively than humans. They can manage data from thousands of sources, including network devices, servers, applications, and other security tools, without compromising performance.
- **Integration and Correlation**: SIEM systems integrate data from multiple sources and correlate events across different systems and time frames. This holistic view enables the identification of sophisticated threats that span multiple vectors and occur over extended periods, which is a challenging task for human analysts without computational assistance.

Continuous Learning

- **Adaptability**: Advanced SIEM systems and AI-based security tools learn over time, adapting to new threats and changing patterns of behavior. Machine learning algorithms can evolve to detect emerging threats faster than humans can disseminate and apply new knowledge.

Comprehensive Coverage

- **Simultaneous Monitoring**: Computers can monitor multiple streams of data simultaneously across various systems and locations. This capability ensures that no part of the network is left unchecked, something a team of human analysts might struggle to achieve due to resource constraints.

While humans are essential for setting up, configuring, and interpreting the outputs of computers and SIEM systems, these technologies surpass human capabilities in the direct tasks of querying, monitoring, and analyzing data. The combination of speed, efficiency, accuracy, scalability,

continuous learning, and comprehensive coverage makes computers and SIEM systems indispensable tools in the field of data analysis and cybersecurity.

Humans vs Computers (Sarah Conner has entered the chat)

Humans and computers (including Security Information and Event Management (SIEM) systems) both play crucial roles in the domain of log analysis and threat detection. While computers and SIEMs excel in specific tasks due to their computational capabilities, humans have distinct advantages in areas that require critical thinking, intuition, and context understanding. Here's a detailed look at what humans are better at than computers in the world of logs and detection:

Contextual Understanding

Humans are particularly adept at understanding the broader context of a situation. They can evaluate external factors, historical precedents, and nuanced details that might not be immediately apparent in the raw data. This ability allows humans to discern patterns or anomalies that might not trigger any flags in a purely automated system but could indicate emerging threats or complex, multi-stage attack strategies.

Creativity and Intuition

Humans can think creatively and apply intuition to problem-solving, which is especially valuable in detecting sophisticated cyber threats. They can hypothesize about attackers' motives, potential next steps, and unconventional attack vectors. This capability enables them to anticipate and mitigate attacks that have not been seen before and for which no predefined detection rules exist.

Ethical Considerations and Decision-Making

Humans are capable of considering ethical implications in their decision-making processes. When analyzing logs and detecting potential threats, humans can weigh the privacy implications, legal boundaries, and ethical concerns associated with different actions, such as monitoring user activity or accessing sensitive data. This nuanced approach to decision-making is something computers and algorithms struggle to replicate.

Adaptability and Learning

While machine learning models can adapt to new data to some extent, humans are inherently adaptable learners who can quickly adjust their strategies based on new information, changing tactics, or evolving threat landscapes. They can learn from less structured feedback and experiences, enabling them to recognize and respond to threats in dynamic environments effectively.

Emotional Intelligence

Humans excel in areas requiring emotional intelligence, such as collaborating with team members, communicating findings effectively, and understanding user behavior from a psychological perspective. This ability can be particularly useful in scenarios where understanding user intent or distinguishing between benign and malicious actions is critical.

Complex Problem-Solving

Humans are skilled at solving complex problems that require the integration of diverse types of information, drawing from various sources and experiences. They can navigate ambiguities and uncertainties in data, making informed decisions even when information is incomplete or contradictory.

While computers and SIEM systems are unparalleled in processing vast amounts of data, performing repetitive tasks, and executing queries at high speed, humans bring critical thinking, creativity, ethical judgment, and the ability to understand complex, multi-faceted contexts to the table. The optimal approach in log analysis and threat detection is a synergistic one, leveraging the strengths of both humans and technology to enhance overall security posture.

HANDS ON KEYBOARD Assignments

Prerequisite

1. Power up your Linux Server VM that you have been using previously
2. On your Host OS Navigate via browser to the provided website windows_activity_logs.txt
3. Download the specified file your host OS
4. In your host OS open a terminal / Shell
5. SCP the file to your Linux Server.
    - scp windows_activity_logs.txt <username>@<ip of linux server>:
6. In the Linux VM CLI, validate that the file in your home folder

Exercise 1

Run the following command:

grep user1 windows_activity_logs.txt

It should run successfully, but provide no output. That is because Linux is case sensitive.

The `grep` command in Linux is case-sensitive by default. This means that it distinguishes between uppercase and lowercase letters when performing searches. The query `grep user1 windows_activity_logs.txt` fails to find the intended records because it searches for "user1" with a lowercase "u", while the logs contain "User1" with an uppercase "U". Since "user1" and "User1" are considered different by the case-sensitive nature of `grep`, the command does not match any lines and thus fails to return the expected results.

To successfully find entries for "User1" regardless of case, you can use the `-i` option with `grep`, which makes the search case-insensitive. This way, `grep` will ignore the case difference between uppercase and lowercase letters, treating "User1", "user1", "USER1", etc., as equivalent.

Option 1

```
grep User1 windows_activity_logs.txt
```

Option 2

```
grep -i 'user1' windows_activity_logs.txt
```

Using the `-i` option ensures that `grep` matches and returns lines containing "User1", regardless of how the case is presented in the logs, thereby overcoming the initial issue where the case-sensitive search failed.

Exercise 2

Run the following command:

```
grep User3 windows_activity_logs.txt
```

Exercise 3

Search the file for document1 using the previous commands as a template.

When successful, search the file for document1.docx using the previous commands as a template.

Consider how and why each search and output are different.

Exercise 4

Task 1

Make a second copy of `windows_activity_logs.txt` and name it `windows_activity_logs2.txt`.

In Linux, show via the list command that both files exist in the CLI, and compare their size.

YOU SHOULD KNOW

In Linux, the `*` wildcard is used as a placeholder to represent any number of characters (including none) in file and directory names. It allows users to perform operations on multiple files or directories that match the given pattern.

Example:

ls *.txt

This command lists all files in the current directory that have a `.txt` extension. The `*` wildcard stands in for any sequence of characters, meaning it will match any file name as long as it ends with `.txt`. This is useful for performing operations on multiple files of a certain type without needing to list them all individually.

Task 2

Use the md5sum command along with the wildcard character to generate and compare the MD5 hashes of two files. Determine the significance of identical hashes.

Task 3

Use the `diff` command to compare two files. Understand the implication of receiving no output from this comparison.

In the Linux command line, when you run the `diff` command on two files and it doesn't produce any output, it means that the files are identical. There are no differences between the contents of the two files.

Exercise 5

Use the `grep` command along with a wildcard to search for the term "Spreadsheet.xls" across multiple files, specifying "filename" as a part of the file search pattern.

PROTIP: A wildcard will be involved. Double check Exercise 4.

Exercise 6

Using `grep` with logical operators

To use `grep` for searching two different patterns (variables) within files, you have a few options, depending on whether you want to find lines that match both patterns (logical AND) or either of the patterns (logical OR).

Logical OR (`pattern1` OR `pattern2`)

If you want to find lines that contain either `pattern1` or `pattern2`, you can use the `-E` option with `grep` and separate the patterns with a pipe `|`

Example:

```
grep -E 'pattern1|pattern2' filename
```

Task 1

Find any logs with User2 AND open file and then redirect it to a file called log1.txt

For example, to search for lines containing either "apple" or "banana" in `file.txt`:

```
grep -E 'apple|banana' file.txt
```

Logical AND (`pattern1` AND `pattern2`)

If you want to find lines that contain both `pattern1` and `pattern2`, regardless of the order, you can chain `grep` commands using pipes. This approach filters the output of the first `grep` through the second `grep`.

```
grep 'pattern1' filename | grep 'pattern2'
```

Task 2

Find logs that contain both "User1" and "failed":

```
grep 'User1' windows_activity_logs.txt | grep 'failed'
```

But then append it to log1.txt

Exercise 7

Use the `grep` command along with a wildcard to search for User1 and failed multiple files, specifying "filename" as a part of the file search pattern from the file you downloaded and then copied previously.

Save output of the above command into a file called log2.txt