Stick to the Script

Cybersecurity

Terminal 101 Day 3

Class Objectives

By the end of today's class, you will be able to:

- Discuss three benefits of text processing programs over programming languages for a security professional.
- Use **sed** to make substitutions to a file.
- Use **awk** to isolate data points from a complex log file.
- Edit the contents of a file using **nano**.
- Design an IP lookup shell script by passing arguments.

Activity: Warm-Up

You continue as a security analyst at Wonka Corp.

Thanks to your great work, the authorities have been able to charge Slugworth Corp. The prosecutor has subpoenaed you for a list of additional data to help build their case.

You are tasked with gathering additional information as specified in the subpoena.

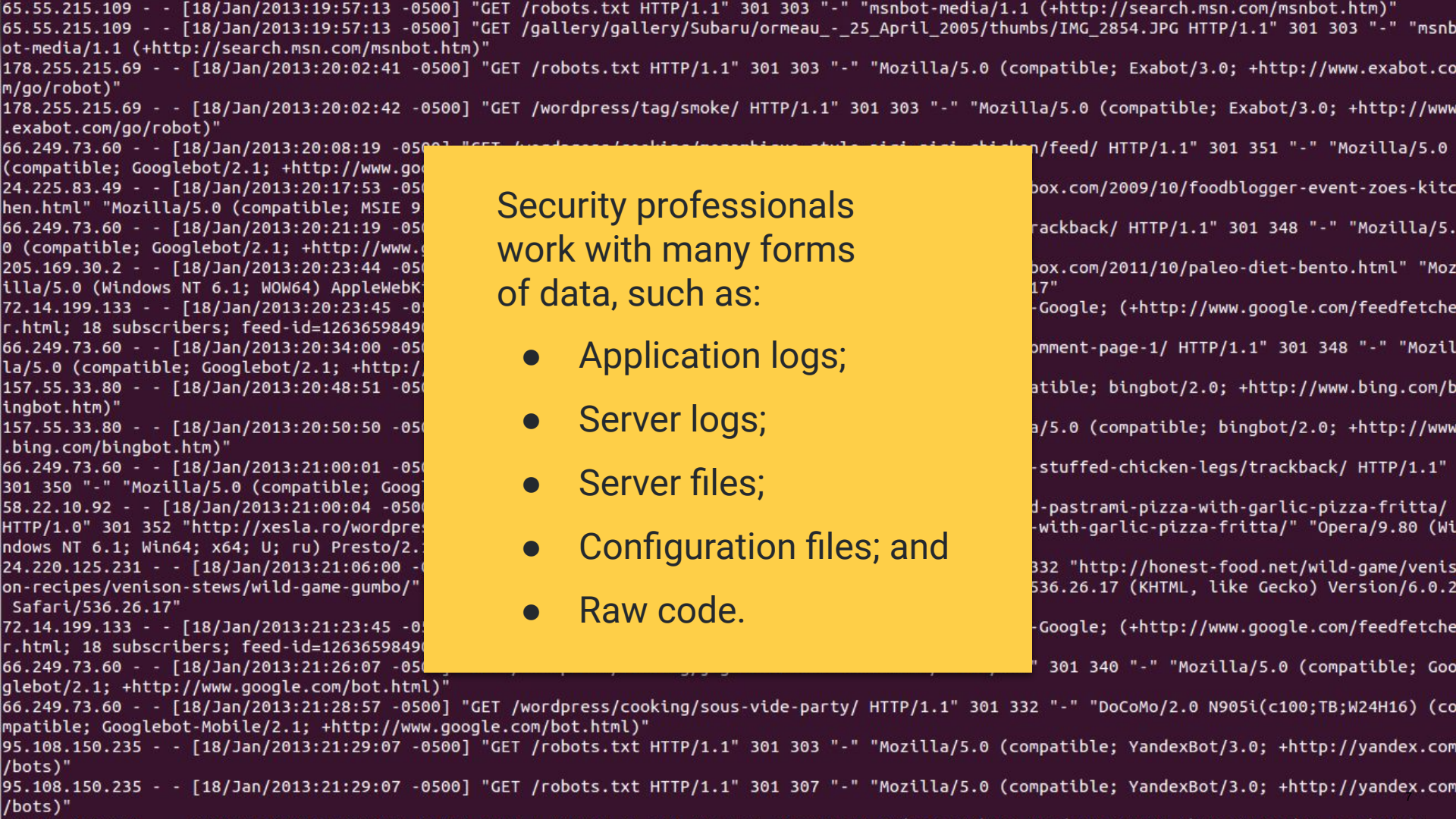
Suggested Time:

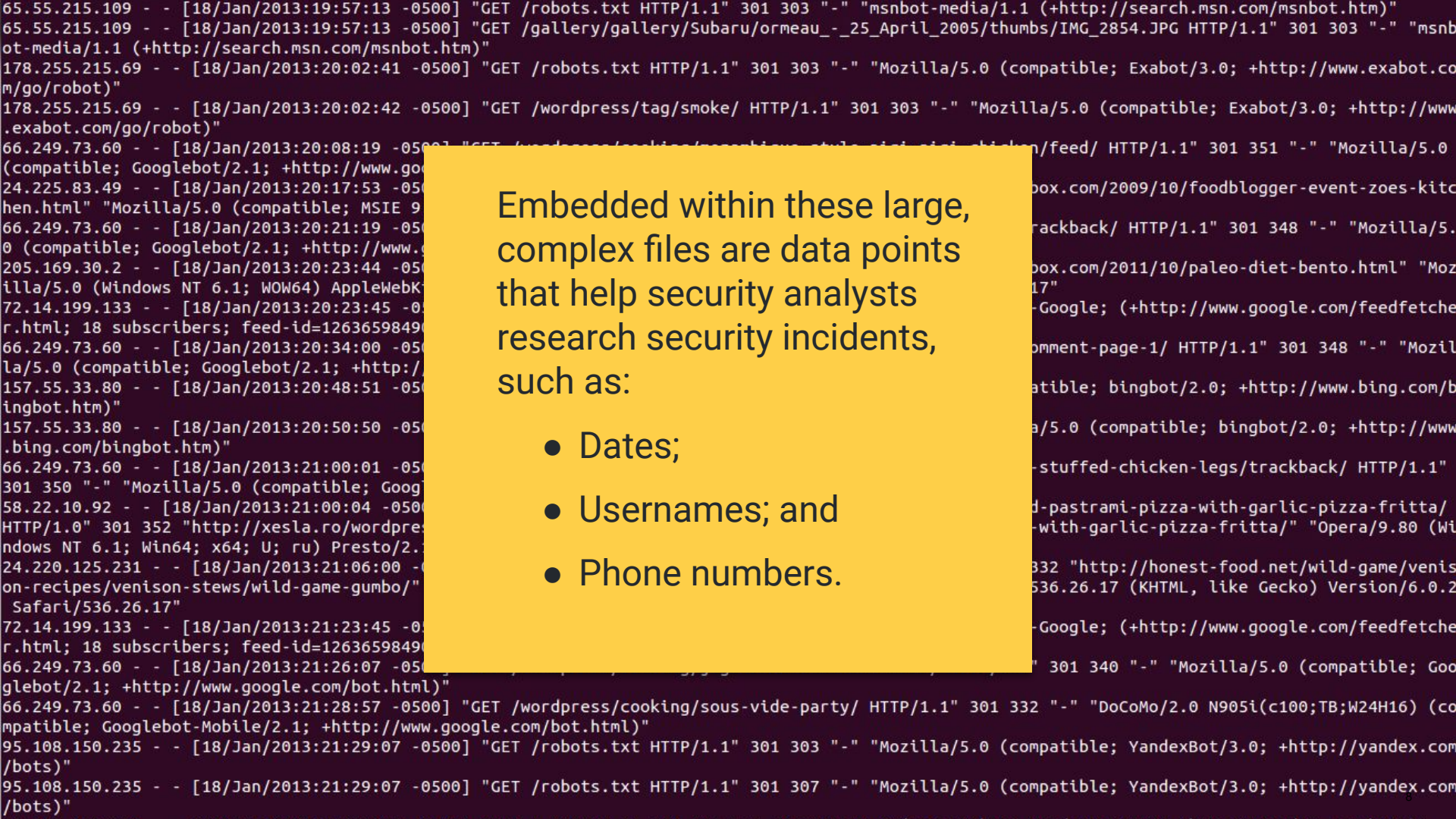
10 Minutes

Time’s Up! Let’s Review.

Questions?

Text Processing


Security professionals work with many forms of data, such as:
Application logs;
Server logs;
Server files;
Configuration files; and
Raw code.

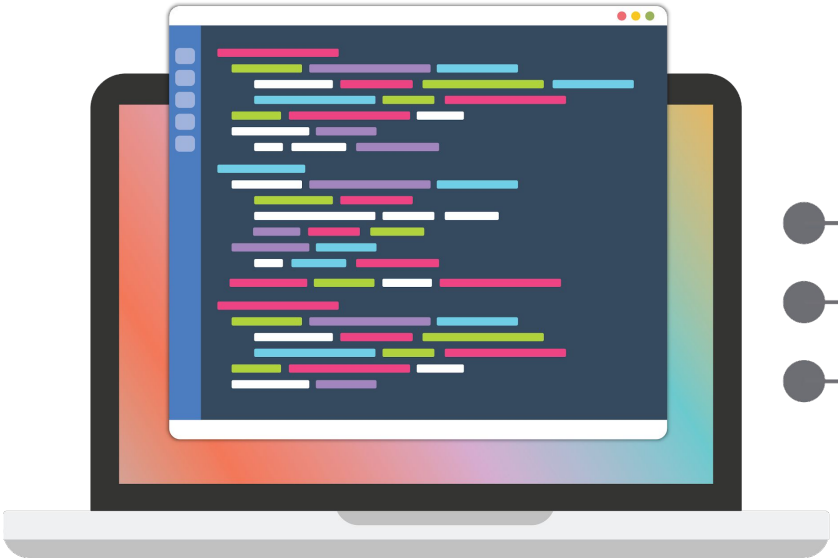

Embedded within these large, complex files are data points that help security analysts research security incidents, such as:
Dates;
Usernames; and
Phone numbers.


We can use this data to:
Isolate attack signatures;
Identify attack vectors; and
Identify timing of attacks.

Text Processing Commands

Text Processing Commands

We can tackle these dense walls of data with text processing commands. These commands manipulate text and simplify complex data by:



- Substituting text
- Filtering lines of text
- Delimiting text

Text Processing Commands

Multiple text processing commands are already built into the terminal, such as:

- Other languages can also manipulate text, such as **Python** and **C++**.
- These are designed for programming, such as for running web applications.

Text Processing Commands

Using text manipulation commands within bash is preferable to using other programming languages for the following reasons:

Simplicity

Text processing is simpler and easier to learn than programming languages.

Convenience

Most text processing commands are pre-installed in terminals, unlike many programming languages.

Compatibility

Text processing commands can easily be added together to complete more advanced tasks—a complicated task with programming languages.

Today we will focus on the powerful text processing utilities of awk and sed.

sed

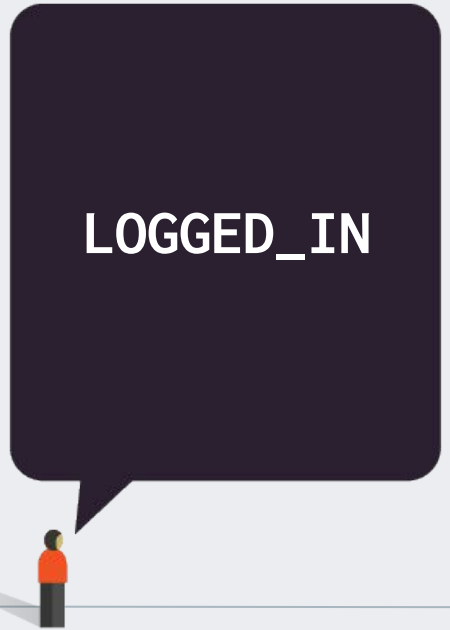
Challenges of Data Processing

Consolidating data from disparate sources can be challenging:

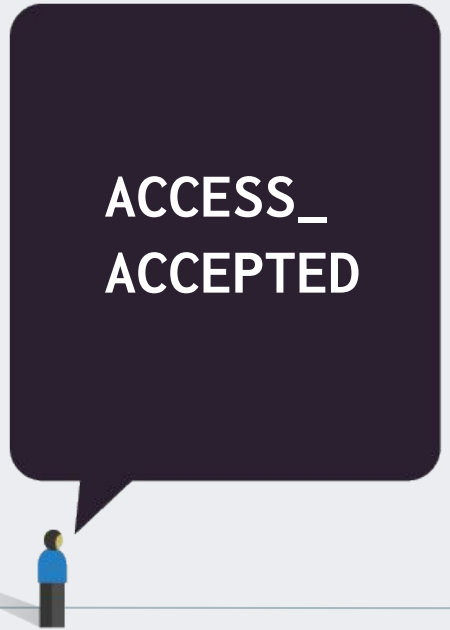
For example:

A security professional wants to analyze login activities from two different applications.

One application labels logins

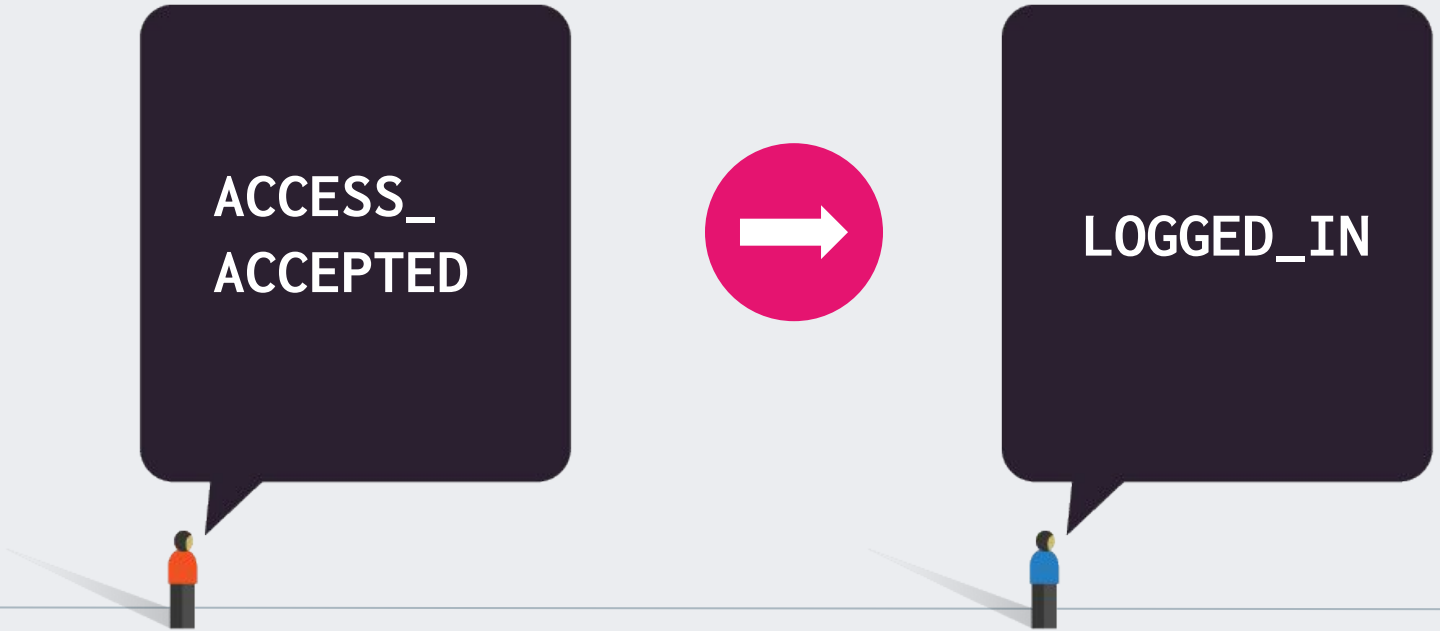


The other labels logins



Challenges of Data Processing

You need to replace the string `ACCESS_ACCEPTED` with `LOGGED_IN` to make the data consistent and allow for both datasets to be analyzed together.



sed

We can solve this problem by using **sed**.

- **sed** stands for **s**tream **ed**itor.
- It reads and edits text line-by-line from an input stream.
- Syntax is: `sed s/(old value)/(replacement value)/`
- It is similar to the familiar “find and replace” feature in Windows systems.

sed

In the next guided tour, we will complete a simple string replacement using **sed**.

```
The Dog Chased the Ball
```

with

```
The Dog Chased the Cat
```

We can use **sed** to replace

`Ball` with `Cat`

Instructor Demonstration

sed

Demo Summary

```
sed s/(old value)/(replacement value)/
```

- **sed** is a text processing utility that can help security professionals analyze data.
- The most basic **sed** capability is string replacement.
- This can help with research by making disparate data sources more consistent.

Activity: Using `sed`

You continue in your role as security analyst at Wonka Corp.

- Your manager suspects a new cybercriminal is attempting to log in to several administrative websites owned by Wonka.
- They provided you with the access logs for two different websites.
- You must combine the two access logs into a single file and use text processing to make the “failed login” data consistent.

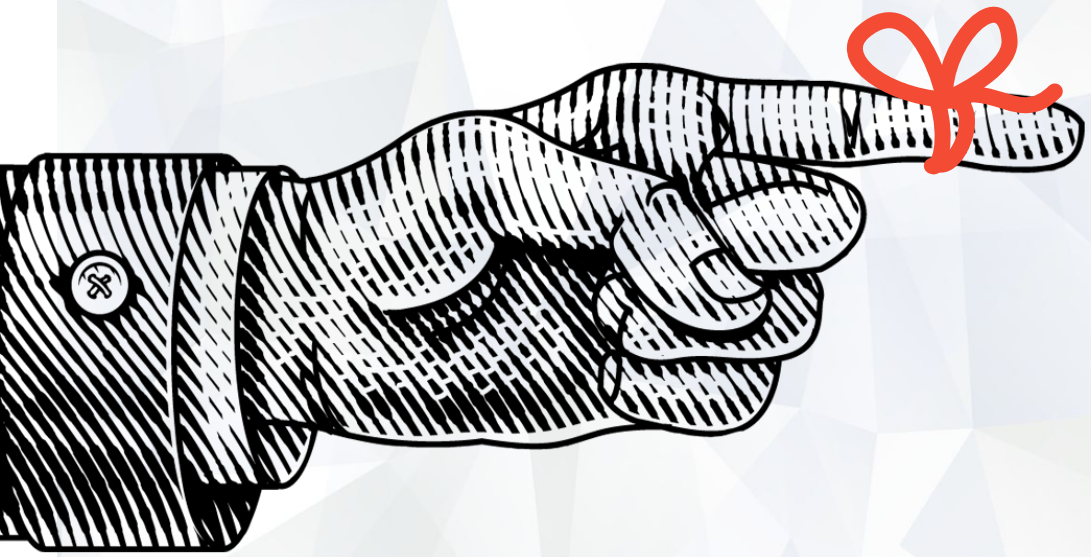
Suggested Time:

10 Minutes

Time’s Up! Let’s Review.

Questions?

awk



Remember,

We'll need to obtain specific data points within complex log files.

Intro to awk

You have a log record containing city, state, and country fields, separated by commas. But, for our analysis, we only need the **state** from the log record.

We have:

```
Atlanta, Georgia, USA
```

But we only need:

```
Atlanta, Georgia, USA
```

Intro to awk

We can technically use **sed** to perform the task:

```
sed 's/^.*\(,.*,\).*$/\1/' | sed 's/,//g'
```

This seems (and is) very complicated.

Fortunately, there is a more efficient tool: **awk**.

Intro to awk

Like **sed**, **awk** is designed for text processing. It is particularly useful for data extraction and reporting against streams of text.

awk is the indicator to tell your operating system to run the **awk** command.

Intro to awk

Like **sed**, **awk** is designed for text processing, but is particularly useful for data extraction and reporting against streams of text.

- `-F`, is the option for doing field separation with **awk**.
- The next value is the **parameter**, which signifies what separates the data in the file.
    - This parameter is also known as the **delimiter**.
- In the example, we used a comma (`,`).

Intro to awk

Like **sed**, **awk** is designed for text processing, but is particularly useful for data extraction and reporting against streams of text.

- **{print $2}'** indicates **awk** is printing the second field, after the fields have been separated by the comma.
- The **$** must be included before the number to indicate the field.
- Multiple fields can be added, with a comma between each.
- The format requires single quotes (`''`) and curly brackets (`{}`)around the print function.

In the next walkthrough, we will isolate the state field from the previous example, using awk.

Demo Scenario: `awk` for a Security Incident

In this demo, we'll act as security analysts at ACME Corp, which just experienced an attack of a high volume of network requests. You received the logs of these network requests.

You must isolate out the **IP addresses** and **time** from these logs to determine which IP address appears most frequently. This IP likely belongs to the attacker.

Instructor Demonstration

Introduction to awk

awk for a Security Incident

Demo Review: Introduction to **awk** & **awk** for a Security Incident

The previous demo covered the following concepts:

01 **awk** is a text processing utility like **sed**, but more robust. It can separate out fields with simple code.

02 Security professionals can use **awk** to isolate specific data points, like IP addresses. This info can help determine what IPs should be blocked.

03 The basic syntax of field separation with **awk** is:
`awk -F(delimiter) '{print $(field_number)}'`

Activity: Using `awk`

You continue in your role as security analyst at Wonka Corp.

- Your manager needs your help finding information on the cybercriminal attempting to log in to administrative websites owned by Wonka.
- You must isolate several fields from the log file to help determine the primary username attempting to log in.

Suggested Time:

10 Minutes

Time's Up! Let's Review.

Questions?


Break

Intro to Shell Scripting

Intro to Shell Scripting

IT and security professionals often have to complete security tasks involving multiple subsequent commands.

Intro to Shell Scripting

For example, an IT professional could be tasked with cleaning up space in a directory. They would need to run the following three commands, in order:

01 **`cd`** to navigate to the directory that contains log files.

02 **`zip`** to compress the files.

03 **`mv`** to move the new zip file into archive folder.

If we script these commands together, they will run, in order, with a single command.

Benefits of Scripting

Scripting provides the following benefits to IT and security professionals:

Efficiency

Running multiple commands with a single script can improve the speed of tasks.

Consistency

Running multiple commands with a single script can improve the consistency of task results.

Reusability

Scripts can be provided to other users for reusability, meaning the same tasks can be completed by someone unfamiliar with the individual commands.

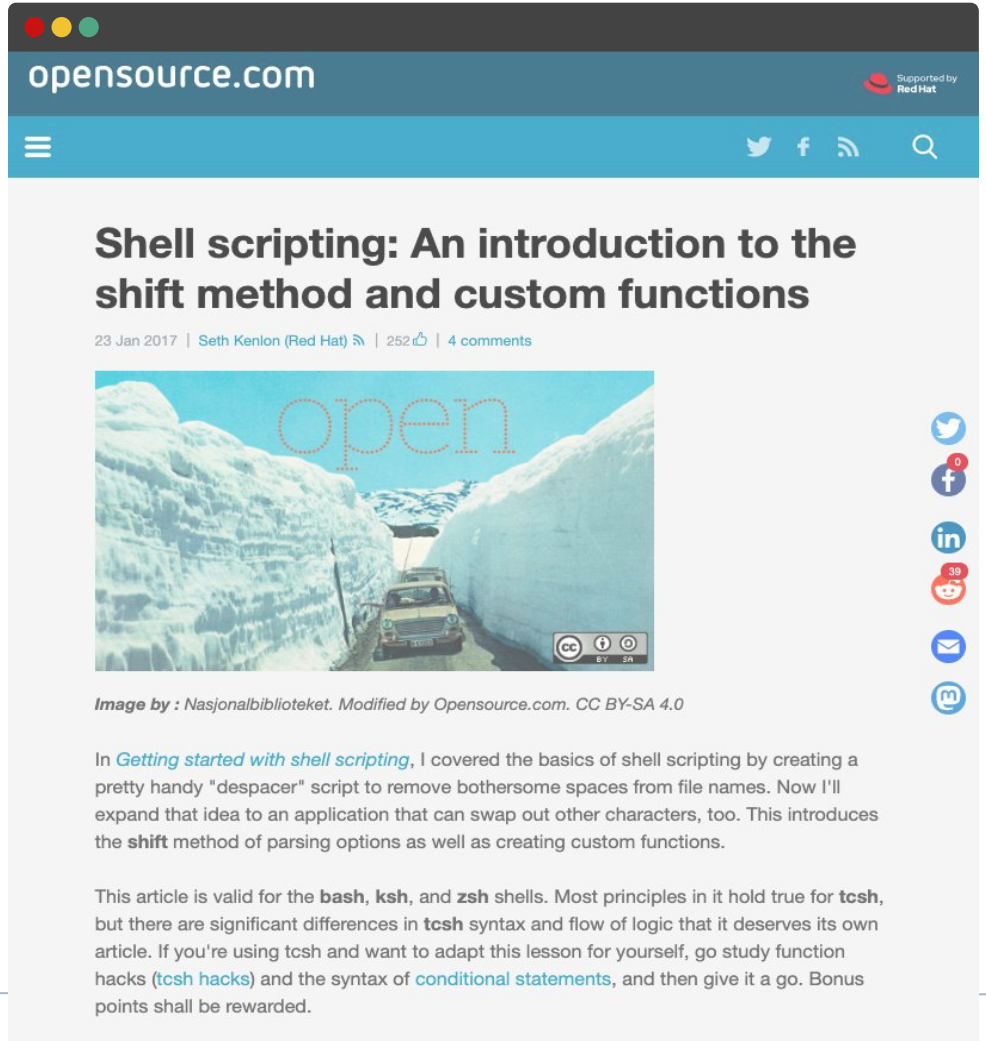
Shell Scripting

We'll focus on a specific type of scripting known as **shell scripting.**

Shell is a terminal program that assists in the interaction between user and the terminal.

Several shells are available in the terminal, such as:

- sh
- bash
- zsh
- csh



(Opensource.com)

Shell Scripting

For example, we'll be using the bash shell. To declare it, we'll write #!/bin/bash.

Let's say you needed to clean up a directory that's running low on space.

For this scenario, we can use a shell script called **`diskcleanup.sh`**. (The extension **`.sh`** is common and often recommended for executable scripts.)

```
01 #!/bin/bash
02 cd log_directory
03 zip logfiles.zip Log*.txt
04 mv logfiles.zip ./archive_directory
```

Shell Scripting (cont.)

To make this script executable we need to run:

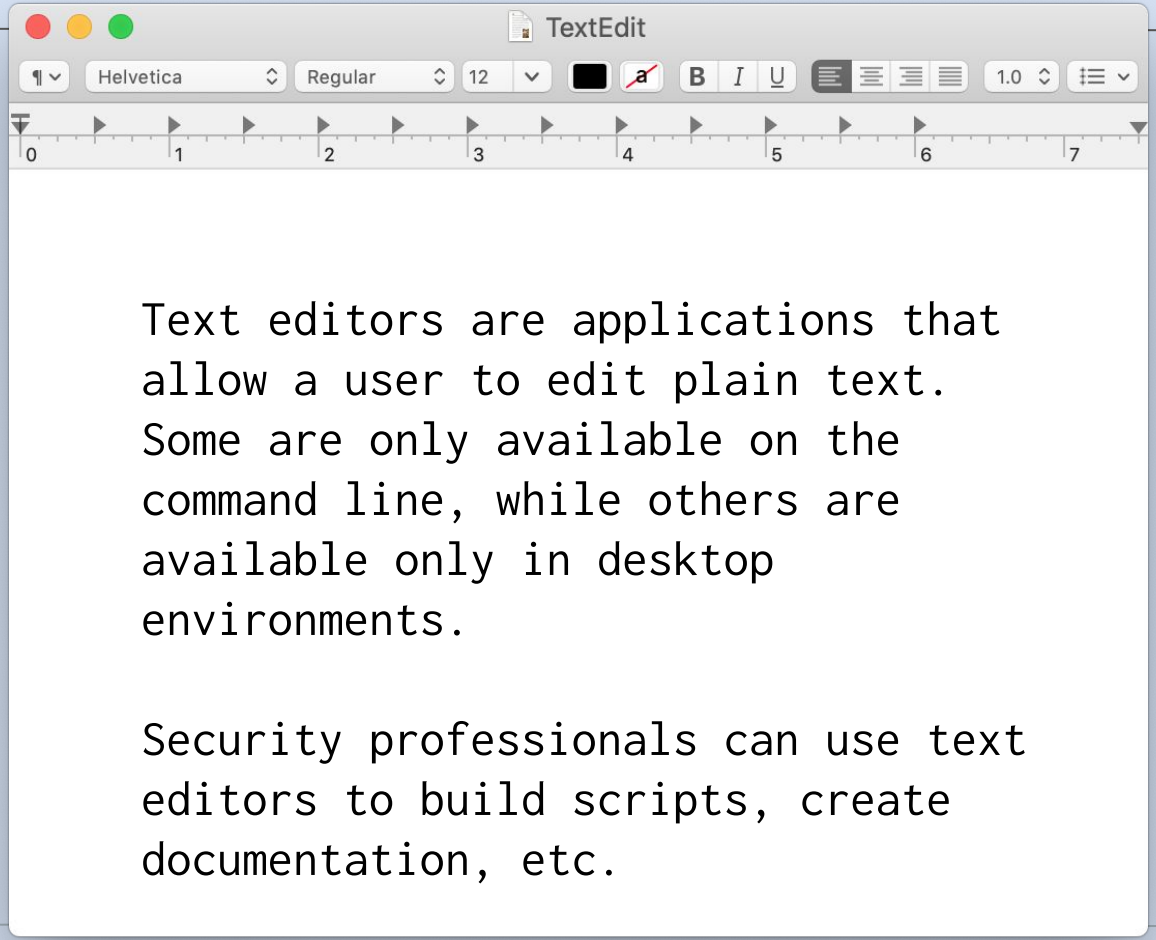
```
chmod +x diskcleanup.sh
```

```
./diskcleanup.sh
```

But wait: To add these commands to our script, we need to know how to write to a file.

Writing and Editing Files

Text Editors



(Source)

Text Editors

You are probably already familiar with text editors.

Notepad on Windows

Text Edit on Mac

Text Editors

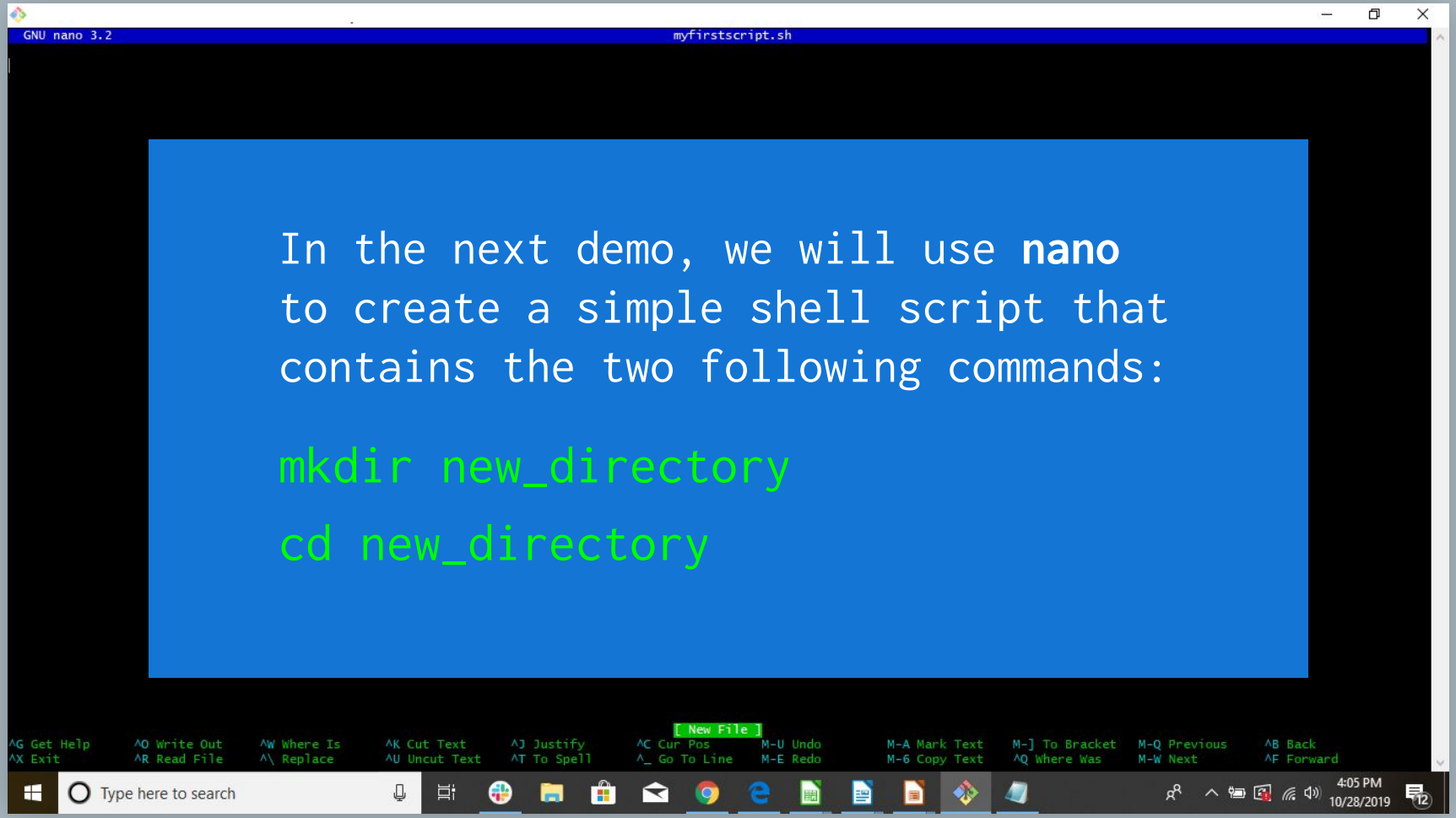
Common text editors available within terminal are: nano, vi, vim, and emacs.

nano

vi

vim

emacs


GNU nano 3.2
myfirstscript.sh
In the next demo, we will use **nano** to create a simple shell script that contains the two following commands:
mkdir new_directory
cd new_directory
[New File]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-Q Previous ^B Back
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text ^Q Where Was M-W Next ^F Forward
Type here to search
4:05 PM
10/28/2019

Instructor Demonstration

My First Shell Script Review

Activity: My First Shell Script

You continue in your role as security analyst at Wonka.

- Your manager needs help creating a simple shell script to automate the `awk` and `sed` tasks you did today on your log file.
- This shell script will be provided to other security analysts so they can easily complete the same tasks.
- You are tasked with using nano to create a shell script with the `awk` and `sed` commands to analyze a log file.

Suggested Time:

15 Minutes

Time's Up! Let's Review.

Questions?

Passing Arguments

Passing Arguments

Sometimes we'll need to change only a single value in a script.

For example:

A security analyst may need to run the cleanup shell script, but only need it to zip up a certain date.

- We need to zip up files named `logs0519`
- We need to change one command in the script:

From `zip logfiles.zip logs*` **To** `zip logfiles.zip logs0519`

Rather than manually editing the script each time it's run, we can add passing arguments.

Passing a Single Argument

When a script is run, you can add an argument by placing it after the script name.

For the previous example:

We could run:

```
sh diskcleanup.sh 0519
```

The syntax is:

```
sh shellscript Argument1
```

Passing a Single Argument

To pass the `0519` into the script, we have to add a `$1` indicating where the first argument, `0519`, will be passed.

We change:

```
zip logfiles.zip logs*
```

To

```
zip logfiles.zip logs$1
```

When the script is run, the command will be changed:

```
sh diskcleanup.sh 0519
```

Will run

```
zip logfiles.zip logs0519
```

Adding Multiple Arguments

Additional arguments can be added with the following syntax:

```
sh shellscript Argument1 Argument2 Argument3
```

For example:

```
sh diskcleanup.sh 0519 backupfolder
```

Within the script, the arguments are represented by:

Argument 1 = `$1`

Argument 2 = `$2`

Activity: Building an IP Lookup Tool

In this activity, you continue in the role of security analyst at Wonka.

- Your manager needs to know where attacks are coming from so that they can block the country from accessing their network.
- Your manager is impressed with the great work you did on the last script and asked you to design a script to identify the countries of IP addresses found in the logs.

Suggested Time:

15 Minutes

Time's Up! Let's Review.

Questions?


The End