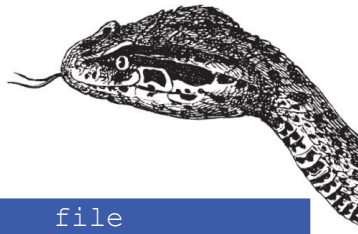
Cybersecurity Ops with bash

Attack, Defend, and Analyze from the Command Line



grep

Search the contents of files

-c	Count matching lines
-E	Enable extended regex
-i	Ignore case
-P	Enable Perl regex
-R	Recursively search

find

Search the system for files

-exec	Execute specified command for each file found
-name	Search by filename
-size	Search by file size
-type	Search by file type

file

Identify file type by magic number

-f	Read list from specified file
-k	List all type matches
-z	Look inside compressed files

Regular Expressions

Character	Meaning
.	Single wildcard character
?	Preceding item is optional
*	Match the preceding item zero or more times
+	Match the preceding item one or more times
^	Anchor pattern to the beginning of the string
$	Anchor pattern to the end of the string
[ ]	Character classes and ranges
( )	Group
{ }	Quantifier

cut

Extract portions of data from a file

-c	Character(s) to extract
-d	Field delimiter
-f	Field(s) to extract

head

Output the first few lines/bytes of file

-n	Number of lines to output
-c	Number of bytes to output

tail

Output the last few lines of a file

-f	Continuously monitor end of file
-n	Number of lines to output

uniq

Remove duplicate lines from a file

-c	Print number of times line is repeated
-f	Ignore the specified number of fields
-i	Ignore case

curl

Network data transfer

-A	Specify user agent
-d	Send using HTTP POST
-G	Send using HTTP GET
-I	Only fetch header
-L	Follow redirects
-s	Do not show errors

sort

Order the lines of a file

-r	Sort in descending order
-f	Ignore case
-n	Use numerical ordering
-k	Sort based on key
-o	Write output to file

join

Combine two files

-j	Join using specified field
-t	Field delimiter

vi commands

b	Back one word
cc	Replace current line
cw	Replace current word
dw	Delete current word
dd	Delete current line
w	Forward one word
:q!	Quit without save
:wq	Quit with save
/	Search forward
?	Search backward
n	Find next occurrence

xxd

Display file in binary or hexadecimal

-b	Display using binary rather than hex
-l	Print specified number of bytes
-s	Start printing at specified position

sdiff

Compare two files

-a	Treat files as text
-i	Ignore case
-s	Suppress common lines
-w	Max characters to output per line

tr

Translate one character to another

-d	Delete character
-s	Squeeze repeated characters

wevtutil

View and manage Windows logs

el	Enumerate available logs
qe	Query a log's events
/c	Specify max number of events
/f	Format output as XML
/rd	Read direction, if true read most recent first

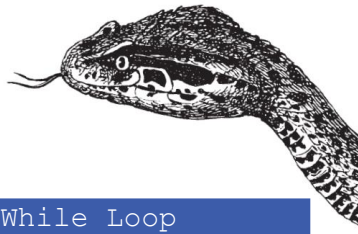
base64

Encode/decode data using Base64

-d	Decode

Cybersecurity Ops with bash

Attack, Defend, and Analyze from the Command Line



Output

Writing to the screen

```
echo 'Hello World'

printf 'Hello World\n'
```

Format Strings

Format strings for `printf`

`%s`	String
`%d`	Decimal
`%f`	Floating point
`%x`	Hexadecimal
`\n`	Newline
`\r`	Carriage return
`\t`	Horizontal tab

Positional Parameters

Script parameters

`$#`	Number of parameters
`$0`	Name of the script
`$1`	First parameter
`$2`	Second parameter …

Default parameters

```
MYVAR=${1:-Cake}
```

Note: If parameter 1 is unset, the value of `MYVAR` will default to `Cake`

User Input

Read from stdin

```
read MYVAR
```

Prompting

```
read -p 'Name: ' USRNAME
```

Reading a File

```
while IFS="" read MYLINE
do
  echo "$MYLINE"
done < "somefile.txt"
```

Note: `IFS=""` preserves whitespace

Variables

Declaring a Variable

```
MYVAR='Hello'
```

Referencing a Variable

```
echo $MYVAR

echo "$MYVAR World"
```

Assigning Shell Output

```
CMDOUT=$(pwd)
```

If Statements

Command conditional (`cmd` will return `0` if success)

```
if cmd
then
  some cmds
else
  other cmds
fi
```

File and numeric conditionals

```
if [[ -e $FILENAME ]]
then
  echo $FILENAME exists
fi
```

File Test	Use
`-d`	Directory exists
`-e`	File exists
`-r`	File is readable
`-w`	File is writable
`-x`	File is executable

Numeric Test	Use
`-eq`	Equal
`-gt`	Greater than
`-lt`	Less than

While Loop

```
i=0
while (( i < 1000 ))
do
  echo $i
  let i++
done
```

For Loop

Numerical looping

```
for ((i=0; i < 1000; i++))
do
  echo $i
done
```

Iterating over a list

```
for VAL in 20 3 dog 7
do
  echo $VAL
done
```

Case Statement

```
case $MYVAR in
  "carl")
    echo 'Hi Carl!'
    ;;
  "paul")
    echo 'Hi Paul!'
    ;;
  *)  # default
    echo 'Goodbye'
    exit
    ;;
esac
```

Functions

Declaring a function

```
function myfun ()
{
  # function body
  echo 'This is myfun()'
}
```

Invoking a function

```
myfun param1 param2
```