12.6.4 Shells and Scripting Facts

A shell refers to the mechanism that allows you to interact with the operating system directly. You enter shell commands in a terminal window and the system responds. Scripting allows you to create virtual programs that automate repetitive tasks. These programs contain statements, commands, and logic. They are often used to start processes, perform backups, and complete system maintenance.

This lesson covers the following topics:

- SSH (Secure Shell)
- OpenSSL
- Scripting environments

SSH (Secure Shell)

The SSH (Secure Shell) protocol is used to secure and encrypt the connection between a client and a server, or other remote device. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network. To use SSH, you need an SSH terminal emulator such as PuTTY.

PuTTY is an SSH and Telnet client, developed originally for the Windows platform. PuTTY is open source software that is developed and supported by a group of volunteers. PuTTY is just one example of a terminal emulator.

SSH can be used for many applications across different platforms including Linux and Windows. SSH can be used for such things as:

- Logging in to a shell on a remote host, Linux, firewalls, and other network device.
- Executing commands on a remote host.
- Setting up automatic (passwordless) login to a remote server.
- Securing file transfer protocols.

OpenSSL

SSL is a method that provides an encryption standard that's widely used by internet websites. When you connect to a secure website, such as a financial institution or shopping site, that site is protected by SSL. Remote access, including PuTTY, can take advantage of the same encryption standard by using OpenSSL, which is an open-source implementation. OpenSSL creates a key pair using encryption standards such as DSA or RSA.

Scripting Environments

A scripting or script language is a programming language for a special run-time environment that automates the execution of tasks. The term scripting language often refers to programs, such as Bash, PowerShell, and Python.

Scripting is often used in software applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, network tasks, as well as numerous other applications. Scripts can be thought of as a small program up to a few thousand lines of code.

As a Security Analyst, you must be familiar with the different scripting languages and recognize the differences.

The table below identifies comparisons in Bash, Python, and PowerShell scripting languages.

Comparison	Bash	Python	PowerShell
Equal	-eq	==	eq
Not Equal	-ne	!= or <>	ne
Greater Than	-gt	>	gt
Greater or Equal	-ge	>=	ge
Less Than	-lt	<	lt
Less or Equal	lle	<=	le

Like all programming languages, there are many common components of each, such as being able to create variables, constants, arrays, and input comments. The table below defines some common programming terms.

Term	Definition
Comment	A *comment* is text in a program's code, script, or another file that is not meant to be used by the program or seen by the user running the program. However, it is seen when viewing the source code.
Variable	A *variable* is a named unit of data that is assigned a value. If and when the value is modified, the name does not change.
Constant	A *constant* is data or a value that does not change, unlike a variable.
Array	An *array* is a group of related data values, or elements, that are grouped together. All the array elements must be the same data type.
If statement	An *if* statement is a conditional statement that, if proven true, performs a function or displays information.
If else statement	An *if else* statement is a conditional statement that selects the statements to run depending on whether an expression is true or false.
Else	*Else* is a conditional statement that if previous conditions are not true displays alternate information or performs alternate commands.
Else if	*Else if*, is a conditional statement performed after an if statement that, if true, performs a function.
String	A *string* is a sequence of characters, either as a literal constant or as some kind of variable.
Substring	A *substring* is a prefix or suffix of any string. For example, a substring of the word computer could be: puter or comp

	could be: puter or comp.

The table below compares code samples of Bash, Python, and PowerShell.

Concept	Bash	Python	PowerShell
Comment	# This line is commented out	# This line is commented out	# This line is commer out
Variables	FirstName = Dana	FirstName = Dana	$FirstName = Dana
Constants	InterestRate = 3.5%	InterestRate = 3.5%	$InterestRate = 3.5%
Basic Arrays	numArray = (num1, num2, num3)	numArray = (num1, num2, num3)	$numArray = @(num num2, num3)
Fetch data stored in Arrays	numArray = (num1, num2, num3) echo {$numArray[0]}	numArray = (num1, num2, num3) print {$numArray[0]}	$numArray = @(num num2, num3) echo $numArray[0]
IF	if [@Cost-t $Balance] then # call payment function fi	if Cost<Balance: #call payment function	If ($Cost-lt $Balance) payment function }

IF, ELSEIF, ELSE	if [<example>] then
	# result
	elif[<example>] then
	# result
	else
	# result
	fi
		if <example>:
		# result
		elif<example>:
		# result
		else:
		# result
			if (<example>){
			# result
			} elseif(<example>) {
			# result
			} else {
			}

Loops	while [$x -eq $dog] do # commands done	animals = ["fox", "wolf", "dog"] for x in animals: print(x) if x == "dog"; break	Do { # commands } While ($x-eq $dog)
String Operations	sampleString = "Sample String" echo ${sampleString} echo ${sampleString:0:4} echo ${sampleString/Sample/Sampling}	sampleString = "Sample String" print sampleString.uppercase print sampleString.replace)Sample", Sampling")	$sampleString - "Sam String" echo $sampleString+ $sampleString.Substr