Bot Security

Level 3- Data Flow Diagram and Threat Model for ServiceNow bot

---

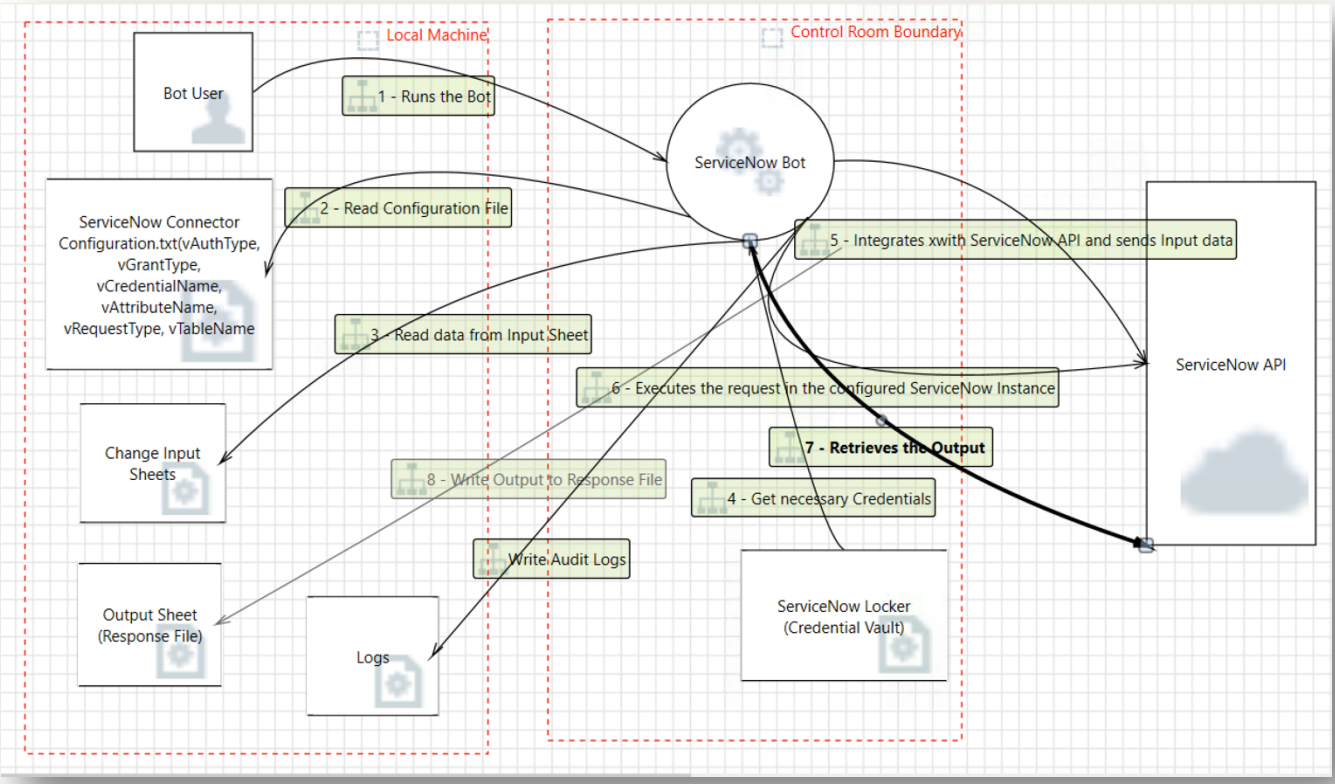
Bot Details

Bot Name: **ServiceNow**
Bot Version/Build: **1.0.0**
Report Prepared by: Satish S – Senior Security Team Lead, Security Innovation
Report Prepared for: **App Perfect**
Date: March 12, 2020

Data Flow Diagram

The ServiceNow bot automates bulk incident, problem & change management in the configured ServiceNow instance. It supports bulk operations like creation, updation, deletion, and retrieval of incidents, problems and change based on the input spreadsheets and generates corresponding output spreadsheets containing the result of the operations. It is a reusable, fast and effective solution to cater most of the needs with least manual intervention.

The following data flow diagram describes the operation and interactions of the **ServiceNow bot** and its associated components.



Data flow Description

The following steps outline the bot's interactions as displayed in the data flow diagram.

1. The ServiceNow bot user invokes the bot to start its processes.

2. The ServiceNow bot reads “ServiceNow Connector Configuration.txt” file (vAuthType, vGrantType, vCredentialName, vAttributeName, vRequestType, vTableName) and decides the “Request Type” based on “vRequestType” field.
3. The bot reads the data from Input Sheet based on “Request Type”.
4. The bot gets necessary credentials from ServiceNow Locker (Credential Vault).
5. The bot connects with ServiceNow API and sends Input Data.
6. The bot executes the request in configured ServiceNow Instance.
7. The bot retrieves the output.
8. The bot writes the output and the error message if any to the “Response File” in “Output” folder. and writes the “Audit Logs”.

List of Identified Threats

The following threats to the **ServiceNow** bot were generated in consultation with the data flow diagram, using the CRUD method (Create, Read, Update and Delete)[1], applying the STRIDE model of threat classification (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege)[2], and other threat brainstorming activities.

[1] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete
[2] https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzZWN1cmVwcm9ncmFtbWluZ3xneDo0MTY1MmM0ZDI0ZjQ4ZDMy

Threat	Description	Asset	Impact
An attacker can read, update or delete data stored in the application binaries	An attacker can read, update or delete data stored in the application binaries.	ServiceNow bot	Information Disclosure, Tampering
Windows user account with excessive privileges	A windows user account with admin privileges are used to run the ServiceNow bot.	ServiceNow bot	Elevation of Privilege
ServiceNow account with excessive privileges	A ServiceNow user account who has access to multiple ServiceNow instances is used.	ServiceNow API	Elevation of Privilege
An attacker can update the business logic in the application scripts	An attacker can modify the business logic in the application scripts and modify the workflow of the bot.	ServiceNow atmx scripts	Tampering
An attacker can create, read or update data between the bot and ServiceNow API	A MiTM attack allows for the modification of traffic between ServiceNow API and ServiceNow bot.	ServiceNow API	Spoofing, Tampering
An attacker can redirect the bot to an invalid ServiceNow API	An attacker can trick the ServiceNow bot to communicate with a fake ServiceNow API resulting in false results.	ServiceNow API	Spoofing
The application performs insufficient input validation	An attacker can provide malformed input resulting in malformed input being processed by bot.	Input File	Repudiation, Denial of Service, Elevation of Privilege
An attacker can create a fake input file	A rogue operating system process, application, or user might create a fake input file and reconfigure the bot to use this file instead. The result would be that output file information would be based on the fake input file, not the real data.	Input File	Spoofing, Tampering, Denial of Service
An attacker can read the input file	An attacker that can read the input file via compromised OS process, application, or user account would gain access to the information stored in input file and could use this information to further launch attacks against known assets.	Input File	Information Disclosure
An attacker can update the input file	An attacker that can update the input file via compromised OS process, application, or user account.	Input File	Denial of Service
An attacker can delete the input file	An attacker that can delete the input file may disrupt the bot from regular operation resulting in the bot being unable to update the input file with accurate information.	Input File	Denial of Service
Insecure Logging	The bot writes sensitive data to logs or snapshots.	Logs	Information Disclosure
An attacker can read the contents of the output file	An attacker that can read the contents of the output file through compromised process, application, or user account would gain access to the output data returned by ServiceNow API	Output File	Information Disclosure
An attacker can update the contents of the output file	An attacker that can update the contents of the output file through compromised process, application, or user account can modify the output returned from ServiceNow API	Output File	Spoofing, Tampering
An attacker can delete the output file	An attacker that can delete the output file might disrupt the execution flow of the bot, resulting in the malfunctioning of bot. It may also result in the loss of historical status data if the data is not being backed up	Output File	Repudiation

Threat Ranking

This section provides the results of the risk analysis performed for the **ServiceNow** bot based on the data flow diagram and identified threats. Threats were reviewed, and the following impact categories were used to identify the risk for each threat: Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability (DREAD)[3].

The Average total is calculated by adding up all the values of the categories and dividing by 5 (number of categories).

Threat	Impact	Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability	Average Total
An attacker can read, update or delete data stored in the application binaries.	Information Disclosure, Tampering	5	8	5	5	9	6.8
Logs stores excessive/sensitive information or logs are not handled properly.	Information Disclosure	6	7	6	7	7	6.6
An attacker can create a fake input file	Spoofing, Tampering, Denial of Service	8	7	5	5	8	6.6
An attacker can update the input file	Tampering	8	6	5	5	8	6.4
An attacker can delete the input file	Denial of Service	7	6	5	5	8	6.2
An attacker can read the contents of the output file	Information Disclosure	7	6	5	5	8	6.2
An attacker can read the input file	Information Disclosure	4	6	5	5	8	5.6
An attacker can modify the business logic in the application scripts and modify the workflow of the bot.	Tampering	8	6	3	5	6	5.6
A windows user account with admin privileges are used to run the ServiceNow bot.	Elevation of Privilege	3	10	5	5	10	5.6
An attacker can update the contents of the output file	Spoofing, Tampering	4	6	5	5	5	5

[3] https://docs.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers#the-dread-approach-to-threat-assessment

Threat	Impact	Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability	Average Total
An attacker can delete the contents of the output file	Repudiation	4	6	5	5	5	5
A MiTM attack allows for the modification of traffic between ServiceNow API and ServiceNow bot.	Spoofing, Tampering, Elevation of Privilege	10	4	3	5	3	5
An attacker can trick the ServiceNow bot to communicate with a fake ServiceNow API resulting in false results.	Spoofing	6	5	3	5	5	4.8
An attacker can provide malformed input resulting in malformed input being processed by bot.	Repudiation , Denial of Service, Elevation of Privilege	7	5	3	5	3	4.6
A ServiceNow user account who has access to multiple ServiceNow instances is used.	Elevation of Privilege	3	5	5	5	5	4.6

Summary of Top Threats

This section summarizes the top threats to the ServiceNow bot and its associated assets. It also combines some of the threats listed is section three into an easily managed grouping. The risk rankings were also re-assessed for the grouped threats.

Threat	Impact	Average Risk Ranking
An attacker can read, update or delete data stored in the application binaries.	Information Disclosure, Tampering	6.8
Logs stores excessive/sensitive information or logs are not handled properly.	Information Disclosure	6.6
An attacker can create a fake input file	Spoofing, Tampering, Denial of Service	6.6

Static Analysis

Bot Details

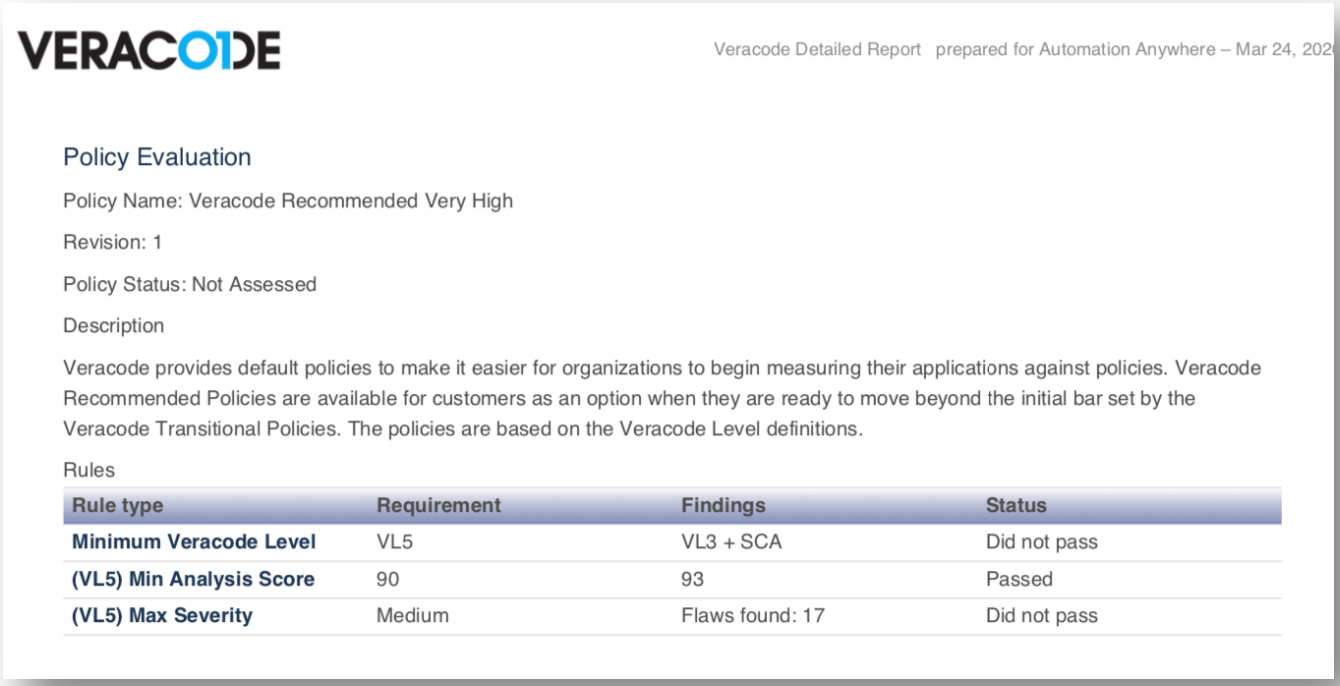
Bot Name: **ServiceNow**
Bot Version/Build: 1.0.0
Report Prepared by: **Satish S – Senior Security Team Lead, Security Innovation**
Report Prepared for: **App Perfect**
Date: March 24, 2020

Application Source Code Scanning

Tool Details

Tool Used: Veracode
Tool version information: VERACODE Engine Version – 20200218164746

Tool Configuration



VERACODE

Veracode Detailed Report prepared for Automation Anywhere – Mar 24, 202

Policy Evaluation

Policy Name: Veracode Recommended Very High

Revision: 1

Policy Status: Not Assessed

Description

Veracode provides default policies to make it easier for organizations to begin measuring their applications against policies. Veracode Recommended Policies are available for customers as an option when they are ready to move beyond the initial bar set by the Veracode Transitional Policies. The policies are based on the Veracode Level definitions.

Rules

Rule type	Requirement	Findings	Status
Minimum Veracode Level	VL5	VL3 + SCA	Did not pass
(VL5) Min Analysis Score	90	93	Passed
(VL5) Max Severity	Medium	Flaws found: 17	Did not pass

Tool coverage

Static Code Analysis was performed on the ServiceNow bot binaries.

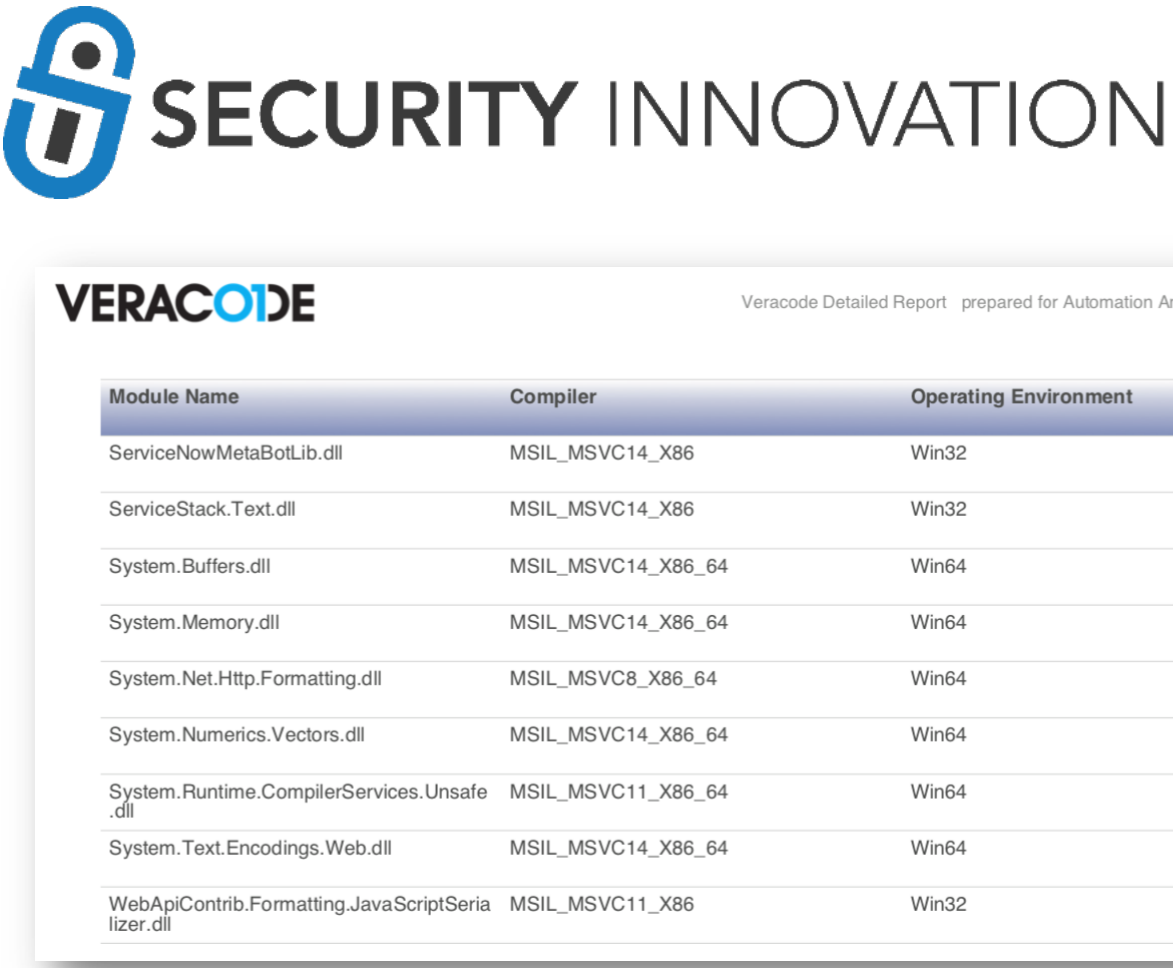
Scope of Static Scan

The following modules were included in the static scan because the scan submitter selected them as entry points, which are modules that accept external data.

Engine Version: 20200218164746

The following modules were included in the application scan:

Module Name	Compiler	Operating Environment	Engine Version
CredentialVaultMetaBotLib.dll	MSIL_MSVC14_X86	Win32	20200218164746
Excel.dll	MSIL_MSVC8_X86_64	Win64	20200218164746
Excel.Helper.dll	MSIL_MSVC8_X86	Win32	20200218164746
ExcelDataReader.DataSet.dll	MSIL_MSVC14_X86	Win32	20200218164746
Microsoft.Extensions.Configuration.Abstractions.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.Configuration.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.Configuration.FileExtensions.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.FileProviders.Abstractions.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.FileProviders.Physical.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.FileSystemGlobbing.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Microsoft.Extensions.Primitives.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Newtonsoft.Json.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
Pomelo.JsonObject.dll	MSIL_MSVC14_X86	Win32	20200218164746



VERACODE

Veracode Detailed Report prepared for Automation Anywhere – Mar 24, 2020

Module Name	Compiler	Operating Environment	Engine Version
ServiceNowMetaBotLib.dll	MSIL_MSVC14_X86	Win32	20200218164746
ServiceStack.Text.dll	MSIL_MSVC14_X86	Win32	20200218164746
System.Buffers.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
System.Memory.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
System.Net.Http.Formatting.dll	MSIL_MSVC8_X86_64	Win64	20200218164746
System.Numerics.Vectors.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
System.Runtime.CompilerServices.Unsafe.dll	MSIL_MSVC11_X86_64	Win64	20200218164746
System.Text.Encodings.Web.dll	MSIL_MSVC14_X86_64	Win64	20200218164746
WebApiContrib.Formatting.JavaScriptSerializer.dll	MSIL_MSVC11_X86	Win32	20200218164746

Unmitigated Vulnerabilities

A total of 20 vulnerabilities were identified during the Static scan.

Flaw Types by Severity and Category

	Static Scan Security Quality Score = 93 (-7) from prior scan			
Very High	**0**			
High	**0**			
Medium	**17**	**(+17)**		
Cryptographic Issues	2	(+2)		
Directory Traversal	4	(+4)		
Encapsulation	1	(+1)		
Information Leakage	8	(+8)		

	Static Scan Security Quality Score = 93 (-7) from prior scan			
Insufficient Input Validation	2	(+2)		
Low	**3**	**(+3)**		
Code Quality	2	(+2)		
Information Leakage	1	(+1)		
Very Low	**0**			
Informational	**0**			
Total	**20**	**(+20)**		

Seventeen Medium issues were identified during the Static scan of the application binaries. The following section describes the medium issues discovered, its severity, description, consequences and recommendations.

Medium (17 flaws) — Fix Required by Policy

Cryptographic Issues(2 flaws)

Description

Applications commonly use cryptography to implement authentication mechanisms and to ensure the confidentiality and integrity of sensitive data, both in transit and at rest. The proper and accurate implementation of cryptography is extremely critical to its efficacy. Configuration or coding mistakes as well as incorrect assumptions may negate a large degree of the protection it affords, leaving the crypto implementation vulnerable to attack.

Common cryptographic mistakes include, but are not limited to, selecting weak keys or weak cipher modes, unintentionally exposing sensitive cryptographic data, using predictable entropy sources, and mismanaging or hard-coding keys.

Developers often make the dangerous assumption that they can improve security by designing their own cryptographic algorithm; however, one of the basic tenets of cryptography is that any cipher whose effectiveness is reliant on the secrecy of the algorithm is fundamentally flawed.

Recommendations

Select the appropriate type of cryptography for the intended purpose. Avoid proprietary encryption algorithms as they typically rely on "security through obscurity" rather than sound mathematics. Select key sizes appropriate for the data being protected; for high assurance applications, 256-bit symmetric keys and 2048-bit asymmetric keys are sufficient. Follow best practices for key storage, and ensure that plaintext data and key material are not inadvertently exposed.

Associated Flaws by CWE ID:

Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327)(2 flaws)

Description

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.

Effort to Fix: 1 - Trivial implementation error. Fix is up to 5 lines of code. One hour or less to fix.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46992	-	10	servicestack.text.dll	string ToMd5Hash(byte[]) 9%	3/24/20
NEW	46991	-	10	servicestack.text.dll	string ToMd5Hash(System.IO.Stream) 9%	3/24/20

Directory Traversal(4 flaws)

Description

Allowing user input to control paths used in filesystem operations may enable an attacker to access or modify otherwise protected system resources that would normally be inaccessible to end users. In some cases, the user-provided input may be passed directly to the filesystem operation, or it may be concatenated to one or more fixed strings to construct a fully-qualified path.

When an application improperly cleanses special character sequences in user-supplied filenames, a path traversal (or directory traversal) vulnerability may occur. For example, an attacker could specify a filename such as "../../etc/passwd", which resolves to a file outside of the intended directory that the attacker would not normally be authorized to view.

Recommendations

Assume all user-supplied input is malicious. Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. When using black lists, be sure that the sanitizing routine performs a sufficient number of iterations to remove all instances of disallowed characters and ensure that the end result is not dangerous.

Associated Flaws by CWE ID:

External Control of File Name or Path (CWE ID 73)(4 flaws)

Description

This call contains a path manipulation flaw. The argument to the function is a filename constructed using untrusted input. If an attacker is allowed to specify all or part of the filename, it may be possible to gain unauthorized access to files on the server, including those outside the webroot, that would be normally be inaccessible to end users. The level of exposure depends on the effectiveness of input validation routines, if any.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Validate all untrusted input to ensure that it conforms to the expected format, using centralized data validation routines when possible. When using black lists, be sure that the sanitizing routine performs a sufficient number of iterations to remove all instances of disallowed characters.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46977	-	1	microsoft.extensions.configuration.fileextensions.dll	Microsoft.Extensions.Configuration.IConfigurationBuilder SetBasePath(Microsoft.Extensions.Configuration.IConfigurationBuilder, string) 97%	3/24/20
NEW	46987	-	7	servicestack.text.dll	string ReadAllText(string) 85%	3/24/20
NEW	46988	-	7	servicestack.text.dll	void CreateDirectory(string) 12%	3/24/20
NEW	46978	-	3	servicenowmetabotlib.dll	void fileCreation(string) 12%	3/24/20

Encapsulation(1 flaw)

Description

Encapsulation is about defining strong security boundaries governing data and processes. Within an application, it might mean differentiation between validated and unvalidated data, between public and private members, or between one user's data and another's.

In object-oriented programming, the term encapsulation is used to describe the grouping together of data and functionality within an object and the ability to provide users with a well-defined interface in a way which hides their internal workings. Though there is some overlap with the above definition, the two definitions should not be confused as being interchangeable.

Recommendations

The wide variance of encapsulation issues makes it impractical to generalize how these issues should be addressed, beyond stating that encapsulation boundaries should be well-defined and adhered to. Refer to individual categories for specific recommendations.

Associated Flaws by CWE ID:

Deserialization of Untrusted Data (CWE ID 502)(1 flaw)

Description

The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46981	-	14	webapicontrib.formatting.javascriptserializer.dll	System.Threading.Tasks.Task<object> ReadFromStreamAsync(System.Type, System.IO.Stream, System.Net.Http.HttpContent, System.Net.Http.Formatting.IFormatterLogger) 50%	3/24/20

→ Information Leakage(8 flaws)

Description

An information leak is the intentional or unintentional disclosure of information that is either regarded as sensitive within the product's own functionality or provides information about the product or its environment that could be useful in an attack. Information leakage issues are commonly overlooked because they cannot be used to directly exploit the application. However, information leaks should be viewed as building blocks that an attacker uses to carry out other, more complicated attacks.

There are many different types of problems that involve information leaks, with severities that can range widely depending on the type of information leaked and the context of the information with respect to the application. Common sources of information leakage include, but are not limited to:

* Source code disclosure
* Browsable directories
* Log files or backup files in web-accessible directories
* Unfiltered backend error messages
* Exception stack traces
* Server version information
* Transmission of uninitialized memory containing sensitive data

Recommendations

Configure applications and servers to return generic error messages and to suppress stack traces from being displayed to end users. Ensure that errors generated by the application do not provide insight into specific backend issues.

Remove all backup files, binary archives, alternate versions of files, and test files from web-accessible directories of production servers. The only files that should be present in the application's web document root are files required by the application. Ensure that deployment procedures include the removal of these file types by an administrator. Keep web and application servers fully patched to minimize exposure to publicly-disclosed information leakage vulnerabilities.

Associated Flaws by CWE ID:

Server-Side Request Forgery (SSRF) (CWE ID 918)(8 flaws)

Description

The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination.

Effort to Fix: 1 - Trivial implementation error. Fix is up to 5 lines of code. One hour or less to fix.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46979	-	2	servicenowmetabotlib.dll	string[] processRequest(string, string, string, string) 56%	3/24/20
NEW	46997	-	8	servicestack.text.dll	System.Net.WebResponse GetResponse(System.Net.WebRequest) 21%	3/24/20
NEW	46996	-	7	servicestack.text.dll	System.Net.WebResponse GetResponse(System.Net.WebRequest) 85%	3/24/20
NEW	46989	-	9	servicestack.text.dll	System.Net.WebResponse PostFileToUrl(string, System.IO.FileInfo, string, string, System.Action<System.Net.HttpWebRequest>) 98%	3/24/20
NEW	46990	-	9	servicestack.text.dll	System.Net.WebResponse PutFileToUrl(string, System.IO.FileInfo, string, string, System.Action<System.Net.HttpWebRequest>) 98%	3/24/20
NEW	46984	-	6	servicestack.text.dll	void MoveNext() 40%	3/24/20
NEW	46985	-	4	servicestack.text.dll	void MoveNext() 53%	3/24/20
NEW	46986	-	5	servicestack.text.dll	void MoveNext() 70%	3/24/20

Insufficient Input Validation(2 flaws)

Description

Weaknesses in this category are related to an absent or incorrect protection mechanism that fails to properly validate input that can affect the control flow or data flow of a program.

Recommendations

Validate input from untrusted sources before it is used. The untrusted data sources may include HTTP requests, file systems, databases, and any external systems that provide data to the application. In the case of HTTP requests, validate all parts of the request, including headers, form fields, cookies, and URL components that are used to transfer information from the browser to the server side application.

Duplicate any client-side checks on the server side. This should be simple to implement in terms of time and difficulty, and will greatly reduce the likelihood of insecure parameter values being used in the application.

Associated Flaws by CWE ID:

Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') (CWE ID 470)(2 flaws)

Description

A call uses reflection in an unsafe manner. An attacker can specify the class name to be instantiated, which may create unexpected control flow paths through the application. Depending on how reflection is being used, the attack vector may allow the attacker to bypass security checks or otherwise cause the application to behave in an unexpected manner. Even if the object does not implement the specified interface and a ClassCastException is thrown, the constructor of the untrusted class name will have already executed.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Validate the class name against a combination of white and black lists to ensure that only expected behavior is produced.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46993	-	11	servicestack.text.dll	object <CreateConstructor>b__0() 91%	3/24/20
NEW	46982	-	12	system.net.http.formatting.dll	object GetDefaultValueForType(System.Type) 91%	3/24/20

Two Low issues were identified during the Static scan of the application binaries. The following section describes the medium issues discovered, its severity, description, consequences and recommendations.

Low (3 flaws)

Code Quality(2 flaws)

Description

Code quality issues stem from failure to follow good coding practices and can lead to unpredictable behavior. These may include but are not limited to:

* Neglecting to remove debug code or dead code
* Improper resource management, such as using a pointer after it has been freed
* Using the incorrect operator to compare objects
* Failing to follow an API or framework specification
* Using a language feature or API in an unintended manner

While code quality flaws are generally less severe than other categories and usually are not directly exploitable, they may serve as indicators that developers are not following practices that increase the reliability and security of an application. For an attacker, code quality issues may provide an opportunity to stress the application in unexpected ways.

Recommendations

The wide variance of code quality issues makes it impractical to generalize how these issues should be addressed. Refer to individual categories for specific recommendations.

Associated Flaws by CWE ID:

Improper Resource Shutdown or Release (CWE ID 404)(2 flaws)

Description

The application fails to release (or incorrectly releases) a system resource before it is made available for re-use. This condition often occurs with resources such as database connections or file handles. Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, it may be possible to launch a denial of service attack by depleting the resource pool.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

When a resource is created or allocated, the developer is responsible for properly releasing the resource as well as accounting for all potential paths of expiration or invalidation. Ensure that all code paths properly release resources.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46980	-	2	servicenowmetabotlib.dll	string[] processRequest(string, string, string, string) 86%	
NEW	46983	-	13	system.net.http.formatting.dll	System.IO.Stream GetStream(System.Net.Http.HttpContent, System.Net.Http.Headers.HttpContentHeaders) 76%	

Information Leakage(1 flaw)

Description

An information leak is the intentional or unintentional disclosure of information that is either regarded as sensitive within the product's own functionality or provides information about the product or its environment that could be useful in an attack. Information leakage issues are commonly overlooked because they cannot be used to directly exploit the application. However, information leaks should be viewed as building blocks that an attacker uses to carry out other, more complicated attacks.

There are many different types of problems that involve information leaks, with severities that can range widely depending on the type of information leaked and the context of the information with respect to the application. Common sources of information leakage include, but are not limited to:

* Source code disclosure
* Browsable directories
* Log files or backup files in web-accessible directories
* Unfiltered backend error messages
* Exception stack traces
* Server version information
* Transmission of uninitialized memory containing sensitive data

Recommendations

Configure applications and servers to return generic error messages and to suppress stack traces from being displayed to end users. Ensure that errors generated by the application do not provide insight into specific backend issues.

Remove all backup files, binary archives, alternate versions of files, and test files from web-accessible directories of production servers. The only files that should be present in the application's web document root are files required by the application. Ensure that deployment procedures include the removal of these file types by an administrator. Keep web and application servers fully patched to minimize exposure to publicly-disclosed information leakage vulnerabilities.

Associated Flaws by CWE ID:

Information Exposure Through Sent Data (CWE ID 201)(1 flaw)

Description

Sensitive information may be exposed as a result of outbound network connections made by the application.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Ensure that the transfer of sensitive data is intended and that it does not violate application security policy or user expectations.

Instances found via Static Scan

	Flaw Id	Module #	Class #	Module	Location	Fix By
NEW	46994	-	7	servicestack.text.dll	void WriteLine(string) 14%	