Passwordless RDP Session Hijacking Feature All Windows versions

Archived: 2026-04-05 20:39:11 UTC

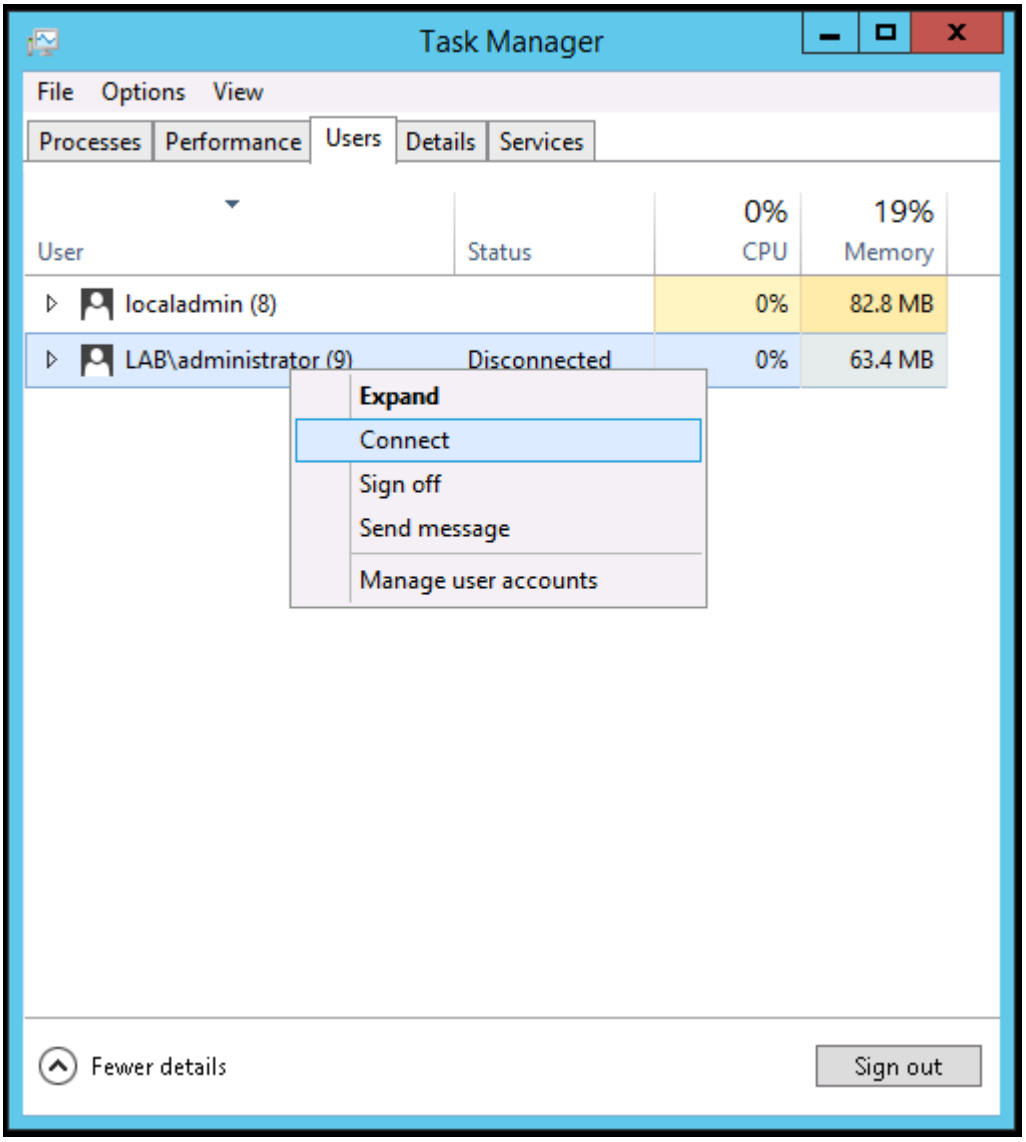
* This post periodically updated, all updates in the end of the post.

Update: Added [Windows Server 2016 Datacenter Demo]

Hey there,

Blogpost in 20 seconds: Fun with sethc backdoored host :) somewhere in the internet:

Recently i've played with sethc/utilman logon screen backdoors, and almost everytime i used just command line. Occasionally i've looked at Users tab in Task Manager (taskmgr.exe), and clicked connect button, and surprisingly **i've got connected to selected user's session**.



When i checked it again with local admin rights, it **failed** by asking user's password.
Why and how that happened? Let's dig deeper.

Related to Microsoft documentation:
https://technet.microsoft.com/en-us/library/cc770988(v=ws.11).aspx
https://technet.microsoft.com/en-us/library/cc731007(v=ws.11).aspx

we can see couple important remarks:

Remarks

- *You must have **Full Control access permission** or Connect special access permission to connect to another session.*
- *The **/dest:**<SessionName> parameter allows you to connect the session of another user to a different session.*

- *If you do not specify a password in the <Password> parameter, and the target session belongs to a user other than the current one,* ***tscon*** *fails* ***(not really)***.

I've got it! Sticky Keys (cmd backdoor) at windows login screen runs with NT AUTHORITY/SYSTEM and have Full Control access permission, and can connect to EVERY user session without asking for a password.

So we've got a session hijacking here. The most funny thing is that the legit user isn't asked for logout, by using this technique the user just will be **kicked out of the session without any notification**.

Attack Vector Details:

A privileged user, which can gain command execution with NT AUTHORITY/SYSTEM rights can hijack any currently logged in user's session, without any knowledge about his credentials.
Terminal Services session can be either in connected or disconnected state.

This is ~~high risk vulnerability~~ which allows any local admin to hijack a session and get access to:

1. Domain admin session.

2. Any unsaved documents, that hijacked user works on.

3. Any other systems/applications in which hijacked user previously logged in (May include another Remote Desktop sessions, Network Share mappings, applications which require another credentials, E-mail etc.)
feature

Example scenario:

Some bank employee have access to billing system, and it's credentials to login.
One day, he come to work, logging in to the billing system and start to work. At lunch time he **will lock his workstation**, and out to lunch.
Then, system administrator gets to employee's workstation, and logs in with his administrator's account.
According to the bank's policy, administrator's account should not have access to the billing system, but with couple of **built-in commands** in windows, this system administrator will hijack employee's desktop which he leaved locked. From now, sysadmin can perform malicious actions in billing system as billing employee account.

There are huge amount of scenarios like this.

Furthermore, an attacker doesn't need to use tools like metasploit, incognito, mimikatz etc, which is commonly used for user's token manipulation and impersonating logged in users. **Everything is done with built-in commands.** Every admin **can** impersonate any logged in user either locally with physical access or remotely via Remote Desktops (see PoC).

Tested on:

Windows 2016 (Confirmed by Kevin Beaumont @GossiTheDog)
Windows 2012 R2

Windows 2008
Windows 10
Windows 7

We can talk about endless amount of examples.

It can be done remotely, as shown in Proof of Concepts.

An attacker can hijack active or disconnected session remotely via remote desktops.
I use this technique about three weeks in my on-going penetration tests on daily basis. It in very simple way helps me to get access to sensitive information like emails, opened documents, clear-text passwords that administrators write down in notepad (not intended for saving, but for temporally writing it somewhere), **opened RDP sessions to another external domains** (think cloud), or another applications that make use of different login credentials.

Someone can say, if you admin, you can dump server's memory and parse it. That's correct, but you don't need it any more. Just two simple commands and you are in. The most incredible thing, is that I don't need to know the credentials of hijacked user, it is pure passwordless hijacking.

A successful **attack heavily related on time** and gathered information. If you need to dump a memory, to get your sensitive info, you're in problem. That means that you've tried all quick-wins that you know.

In example of hijacking user (active or disconnected) while he is working now remotely on some sensitive server that i have no access to, and haven't even knew about it, this technique allows me to compromise that server in **less than a minute**. Everything is real and from my own experience.

Furthermore, as I understand it is very hard to catch if this attack happen. Kevin Beaumont @GossiTheDog make an alert on tscon.exe usage, with Microsoft OMS.

I had a conversation about this finding with Benjamin Delpy @gentilkiwi author of mimikatz:
"That is normal Windows API, that's the design flow, they use it. As mentioned earlier, if you admin, you can do everything. But here is the point. Why and HOW you become admin? If some unprivileged user becomes admin using some kind of local privilege escalation - that's the problem and not the design flow we are talking about. You can do everything, even patch terminal services the way that it will accept your token and allow shadowing mode, without user's knowledge.", he said.

Proof of Concept:

Microsoft documentation helps us to do that from command line:

All we need is NT AUTHORITY/SYSTEM command line.

Easiest method with psexec, but requires psexec.exe to be there:

```
psexec -s \\localhost cmd
```

Another method is to create a service that will connect selected session to ours.

1. Get all sessions information:

```
C:\Windows\system32>query user
 USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
 administrator                             1  Disc            1  3/12/2017 3:07 PM
>localadmin            rdp-tcp#55          2  Active          .  3/12/2017 3:10 PM

C:\Windows\system32>
```

2. Create service which will hijack user's session:

```
C:\Windows\system32>sc create sesshijack binpath= "cmd.exe /k tscon 1 /dest:rdp-tcp#55"
[SC] CreateService SUCCESS
```

3. Start service:

```
net setart sesshijack
```

Right after that your session will be replaced with target session.