"Spoofing credential dialogs on macOS, Linux and Windows"

By wunderwuzzi

Published: 2021-04-19 · Archived: 2026-04-05 14:42:38 UTC

A nifty way for adversaries to acquire passwords during post-exploitation is to spoof credential dialogs and perform a local phishing attack. This means tricking a user on a compromised computer to enter their password.

Unfortunately, users are conditioned to enter their credentials frequently and therefore don’t question random passwords prompts too much.

Long, long time ago… but nothing has changed

The idea to spoof a credential dialog is one of the most simple ideas one might come up with.

In fact, one of the first programs I wrote when learning C, some 26 years ago, was a tool that mimicked the Novell Netware Login screen. I was amazed how simple this could be done.

I was just learning about `printf` and `scanf` and put them to creative use. Obviously, I didn’t use it for nefarious purposes - I was and still am a curious person. But I remember showing it to a few friends and got surprised reactions.

In this post we will go over three scenarios on how an attacker might trick users on Mac, Linux and Windows. Given the commands, we can then also easily build detections for them.

macOS: Spoofing a credential prompt using osascript

On macOS the `osascript` command line utility can be used. For instance this will just create a notification:

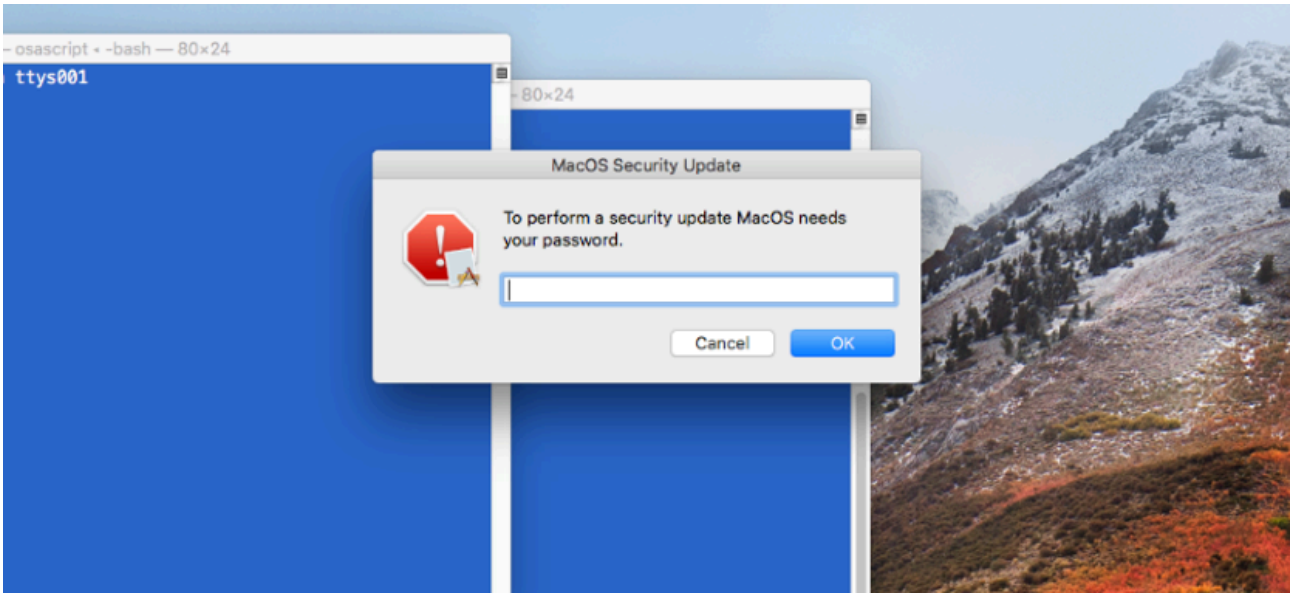
```
# osascript -e 'display notification "Hello World!"'
```

Neat.

It also allows to create a password prompt… There is the display dialog feature with a hidden answer option to create a password prompt.

```
 # PWD_SPOOF=$(osascript -e 'display dialog "To perform a security update MacOS needs your password." with title
# echo $PWD_SPOOF
button returned:OK, text returned:S3cr3tPa$$w0rd!
```

That’s pretty much all that is needed for a post exploit technique.



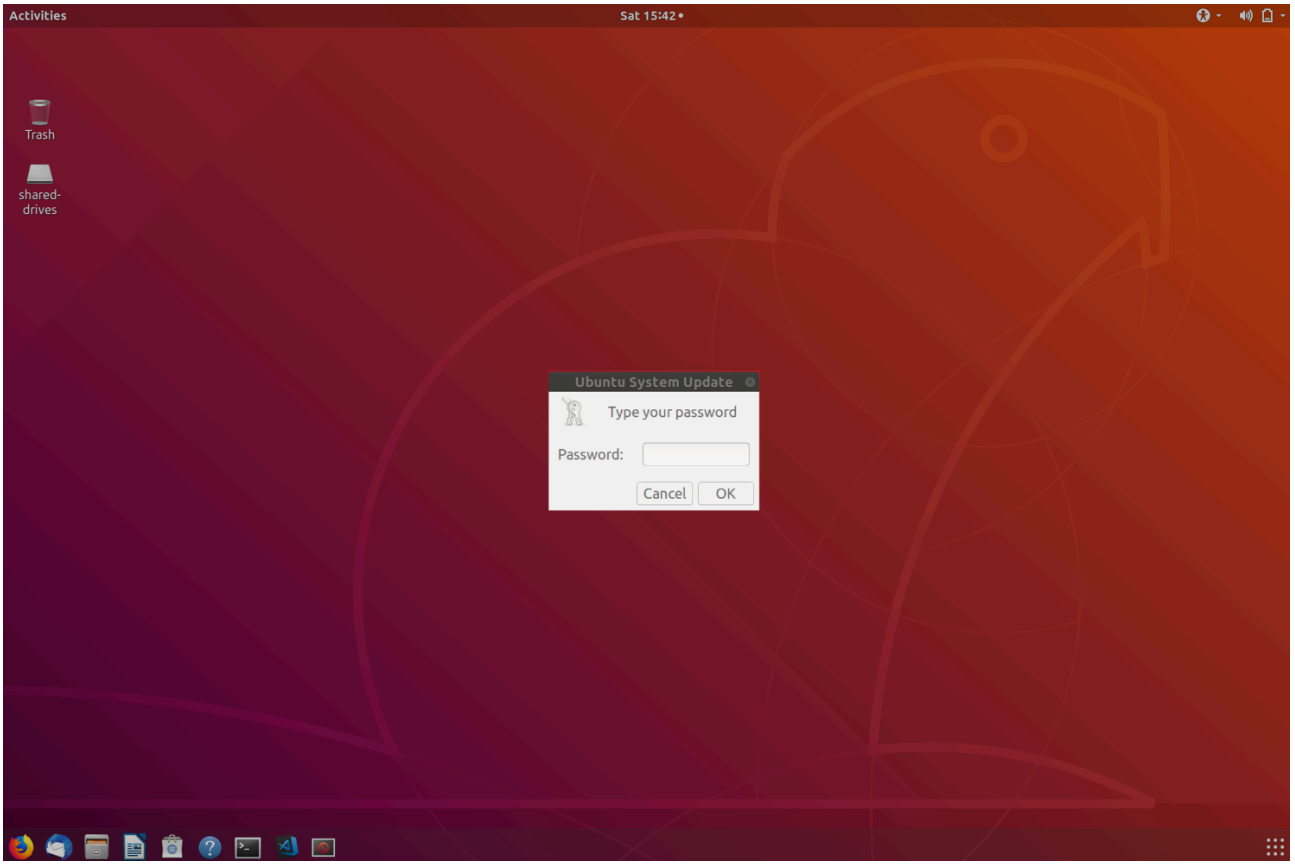
One way to deploy this is by updating the victim's profile.

Linux: Spoofing a credential prompt via zenity

Typically, you won't encounter that many Linux desktop users during red teaming operations.

But if, then `zenity` is useful and it even has a `-password` option that can be used.

```
wuzzi@saturn:$ PWD=$(zenity --password --text "Ubuntu Update needs your password: " --title "Ubuntu System Upda
```



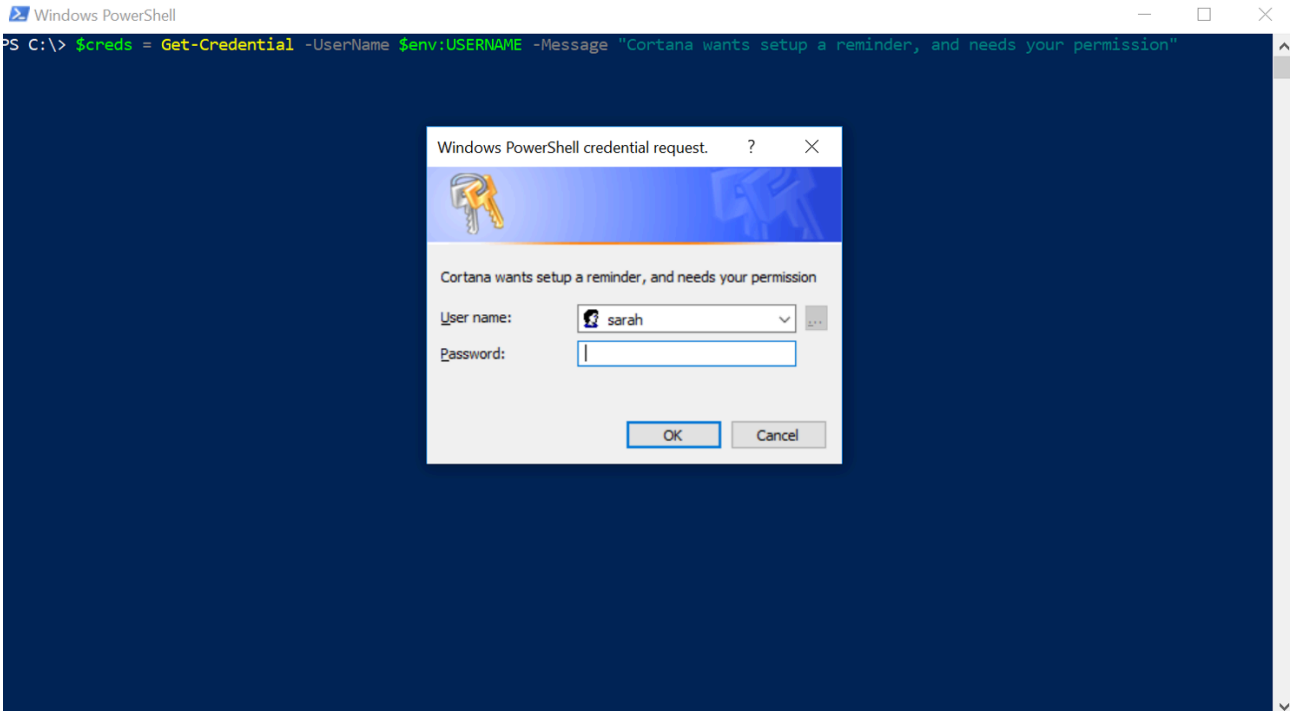
That’s a way I was able to emulate this attack on Linux.

Windows: Spoofing a credential prompt with PowerShell

On Windows `Get-Credential` is your friendly PowerShell command to do this.

```
$creds = Get-Credential -UserName $env:USERNAME -Message "Cortana wants setup a reminder and needs your permis
```

This results in a pop with the following look and feel:



A simple line like this can be added to a user's logon script, AutoStart, or it can be actively injected in another user's session on the machine during post-exploitation.

Detections

By looking through command history one can quickly identify such spoofing attempts. It will take some training and filtering to distinguish legit use cases from attacks, but that's why we have the red team.

Special attention can be given to the `--password` and `with hidden answer` command line arguments on Mac and Linux.

So, make sure that your blue teams has detections for this in place, and is able to distinguish noise from actual attacks.

There are many variations attackers can use, so be on the lookout.

The MITRE ATT&CK framework also covers this via T1411. The details of the TTP have a couple of real world examples of malware doing local phishing attacks.

In Windows one used to have to press `CTRL+ALT+DEL` on the login screen to prevent some of these attacks, but these days it's possible to just spoof the entire login screen by writing a custom credential provider.

Conclusion

This was an overview of three common post-exploitation credential stealing techniques red team and blue teams should be aware of and test for to ensure you can detect adversaries misusing them.

Cheers, @wunderwuzzi23

Also, if you found this interesting check out my book about Red Teaming.
