VB2020
localhost

30 September - 2 October, 2020 / vblocalhost.com

MOST SOPHISTICATED TECHNIQUE OF THE YEAR GOES TO...

Kalpesh Mantri
Quick Heal, India

kalpesh.mantri@quickheal.com

ABSTRACT

As has been the case in the recent past, 2019 was full of new malware campaigns and APT attack discoveries. Some were discovered for the first time, while many made a comeback. We have been tracking such attacks for several years and have observed a variety of techniques being used in them. In this talk we will share a few highly sophisticated techniques used by attackers that have helped the attacks to remain undetected for years. These techniques are not very prevalent at this point; however, we suspect more and more attackers to adopt them in the future.

In this paper I will discuss following:

- An APT actor was found communicating with command-and-control servers over VPN. This APT was able to bypass two-factor authentication (2FA) as well!
- Are password managers safe? Should we use any? They are increasingly being targeted by threat actors to gain credentials.
- How a ransomware group is using the Wake-on-LAN (WoL) feature to increase monetization of infections.
- Web skimmers have started using a formerly retired technique called steganography and are still successfully evading security solutions. Could this technique make a comeback?

I will share insights into the techniques used in these attacks and will discuss the questions called out above. The aim of this paper is to bring these sophisticated techniques to defenders' attention so that we all can work on proactively blocking attacks using them.

INTRODUCTION

Cyber threat actors constantly invest resources in improving their tools and techniques, to stay ahead of the latest security solutions. This means defenders must also evolve and come up with even better ways to tackle cyber attacks. One of the approaches that's catching up in the defender community is to move upwards in the 'The Pyramid of Pain' (introduced by David Bianco [1]) – a simple diagram that shows the relationship between the types of indicators that defenders might use to detect an adversary's activities and how much pain it will cause them when those indicators are denied to them.

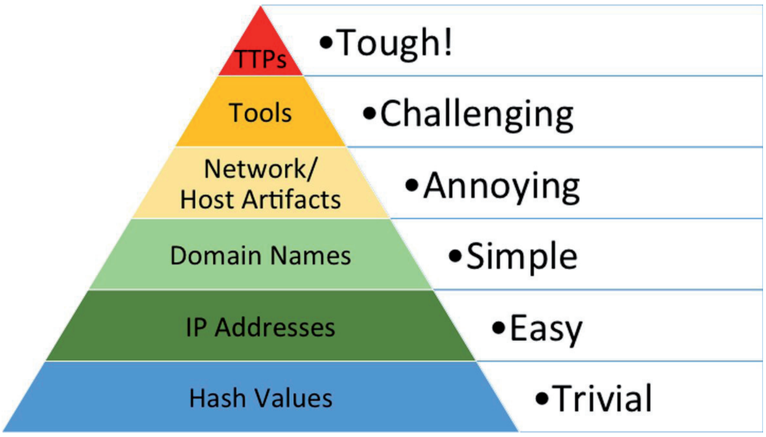


Figure 1: Pyramid of Pain [1].

As per the pyramid, if defenders can prevent, hunt and detect attacks based on the attacker tactics, techniques and procedures (TTPs), then they make it very expensive for the cyber attackers to pivot their path.

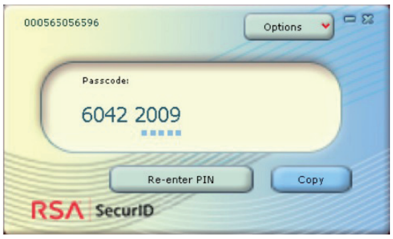
As part of our ongoing tracking of cyber attacks we have been analysing the TTPs used in them. This paper will talk about some sophisticated attack techniques observed during the last year. These techniques are not ubiquitous at this point; however, we believe it's only a matter of time before that happens.

EXFILTRATION VIA VPN WITH TWO-FACTOR AUTHENTICATION

A VPN solution provides a secure communication channel by extending a private network over the public Internet. Almost all VPN solutions today support end-to-end encryption and multi-factor authentication to enhance their security posture.

Most organizations use VPN solutions to enable remote connectivity to on-premise resources. Interestingly, in a recent cyber attack, the threat actor used an organization's VPN channel to exfiltrate data to their servers. They even bypassed the VPN solution's multi-factor authentication mechanism.

In this case, the target organization was using the *RSA SecurID* token generation software to create two-factor codes on endpoints. A *SecurID* token is usually generated for a 'specific system' and is supposed to be tied with that system. But this software had a bug: it checked the 'specific system value' only when importing the *SecurID* token seed, but didn't use it at the time of generating actual two-factor tokens. So, by patching the verification bytes, the software could be made to work on any system.

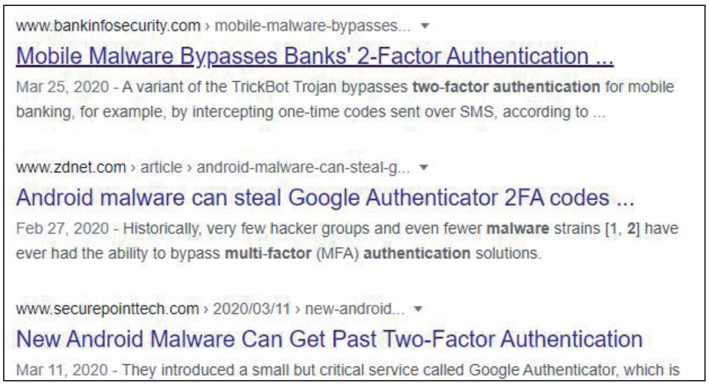


Passcode:
6042 2009

Figure 2: RSA SecurID generating valid two-factor codes.

In this case, the attacker stole an *RSA SecurID* software token and then patched that simple instruction. Afterwards, they were able to generate valid tokens on any machine and use them to exfiltrate data. Using this approach attackers were able to hide their traffic inside the normal VPN traffic of the organization.

This technique seems to be gaining popularity among other groups as well. Recently, a few banking trojans and some *Android* malware have been seen making use of similar techniques.



www.bankinfosecurity.com › mobile-malware-bypasses... ▾

Mobile Malware Bypasses Banks' 2-Factor Authentication ...

Mar 25, 2020 - A variant of the TrickBot Trojan bypasses **two-factor authentication** for mobile banking, for example, by intercepting one-time codes sent over SMS, according to ...

www.zdnet.com › article › android-malware-can-steal-g... ▾

Android malware can steal Google Authenticator 2FA codes ...

Feb 27, 2020 - Historically, very few hacker groups and even fewer **malware** strains [1, **2**] have ever had the ability to bypass **multi-factor** (MFA) **authentication** solutions.

www.securepointtech.com › 2020/03/11 › new-android... ▾

New Android Malware Can Get Past Two-Factor Authentication

Mar 11, 2020 - They introduced a small but critical service called Google Authenticator, which is

Figure 3: 2FA TTP used by recent malware.

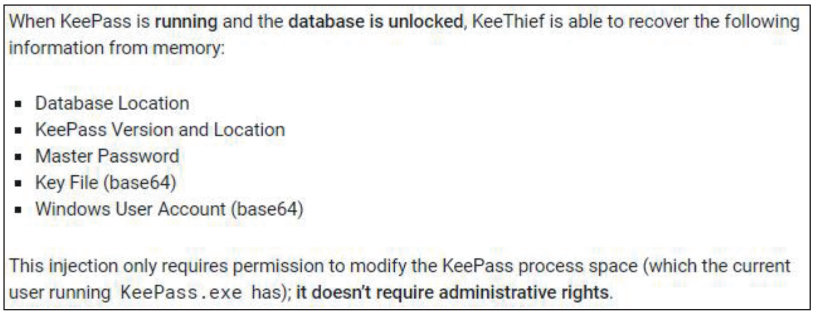
The defender community and organizations should pay attention to this technique as the number of attacks using this mechanism is likely to increase in coming years.

PASSWORD MANAGERS ARE NOT SO SAFE!

Several organizations use password managers to help employees to store passwords safely. *KeePass*, one such password manager tool, was targeted in a recent attack.

In this attack, the attacker retrieved passwords saved in the *KeePass* vault and then used them to infiltrate further inside the organization. This approach completely does away with the need for brute force attempts to break passwords, which can alert the organization's security solutions. With this approach, an attacker could navigate silently and remain undetected for a long period.

Attackers used KeeThief, an open-source PowerShell tool, to recover the master password from the running *KeePass* process. *KeeThief* is, in fact, a popular credentials dumping tool and is available on *GitHub* [2].



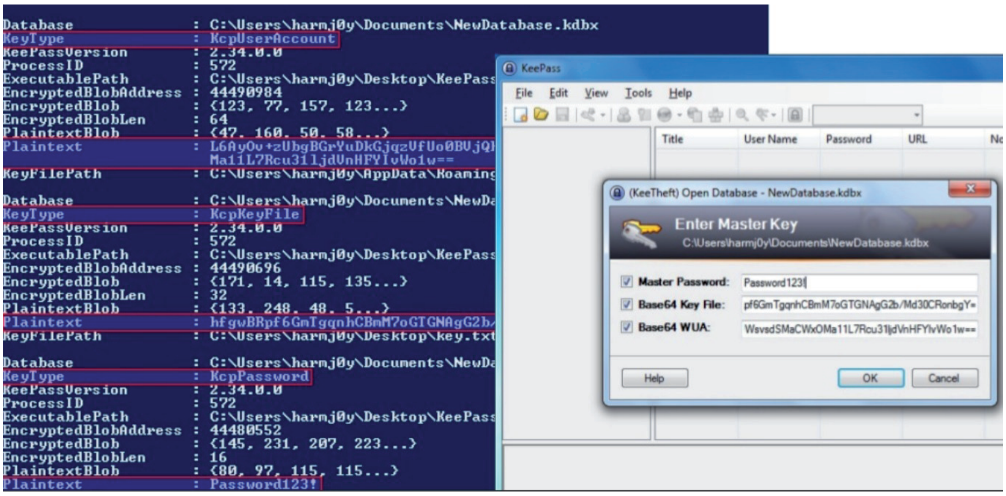
When KeePass is **running** and the **database is unlocked**, KeeThief is able to recover the following information from memory:

- Database Location
- KeePass Version and Location
- Master Password
- Key File (base64)
- Windows User Account (base64)

This injection only requires permission to modify the KeePass process space (which the current user running `KeePass.exe` has); **it doesn't require administrative rights**.

Figure 4: KeeThief functionality.

The attacker apparently took inspiration from Will Schroeder's BSidesNola 2017 presentation [3, 4] and followed similar steps. On target systems, password managers were directly targeted and their contents retrieved.



```
Database              : C:\Users\harmj0y\Documents\NewDatabase.kdbx
KeyType               : KcpUserAccount
KeePassVersion        : 2.34.0.0
ProcessID             : 572
ExecutablePath        : C:\Users\harmj0y\Desktop\KeePass
EncryptedBlobAddress  : 44490984
EncryptedBlob         : {123, 77, 157, 123...}
EncryptedBlobLen      : 64
PlaintextBlob         : {47, 160, 50, 58...}
Plaintext             : L6AyOv+zUbgBGrYuDkGjqzVfUo0BVjQ
                        Ma11L7Rcu31ljdVnHFYIvWo1w==
KeyFilePath           : C:\Users\harmj0y\AppData\Roaming

Database              : C:\Users\harmj0y\Documents\NewDa
KeyType               : KcpKeyFile
KeePassVersion        : 2.34.0.0
ProcessID             : 572
ExecutablePath        : C:\Users\harmj0y\Desktop\KeePass
EncryptedBlobAddress  : 44490696
EncryptedBlob         : {171, 14, 115, 135...}
EncryptedBlobLen      : 32
PlaintextBlob         : {133, 248, 48, 5...}
Plaintext             : hfgwBRpf6GmTgqnhCBmM7oGTGNAgG2b/
KeyFilePath           : C:\Users\harmj0y\Desktop\key.txt

Database              : C:\Users\harmj0y\Documents\NewDa
KeyType               : KcpPassword
KeePassVersion        : 2.34.0.0
ProcessID             : 572
ExecutablePath        : C:\Users\harmj0y\Desktop\KeePass
EncryptedBlobAddress  : 44480552
EncryptedBlob         : {145, 231, 207, 223...}
EncryptedBlobLen      : 16
PlaintextBlob         : {80, 97, 115, 115...}
Plaintext             : Password123!
```

(KeeTheft) Open Database - NewDatabase.kdbx

Enter Master Key
C:\Users\harmj0y\Documents\NewDatabase.kdbx

☑ Master Password: Password123!
☑ Base64 Key File: pf6GmTgqnhCBmM7oGTGNAgG2b/Md30CRonbgY=
☑ Base64 WUA: WsvsdSMaCWxOMa11L7Rcu31ljdVnHFYIvWo1w==

Figure 5: Recovering the plaintext master password for KeePass DB (source: [4]).

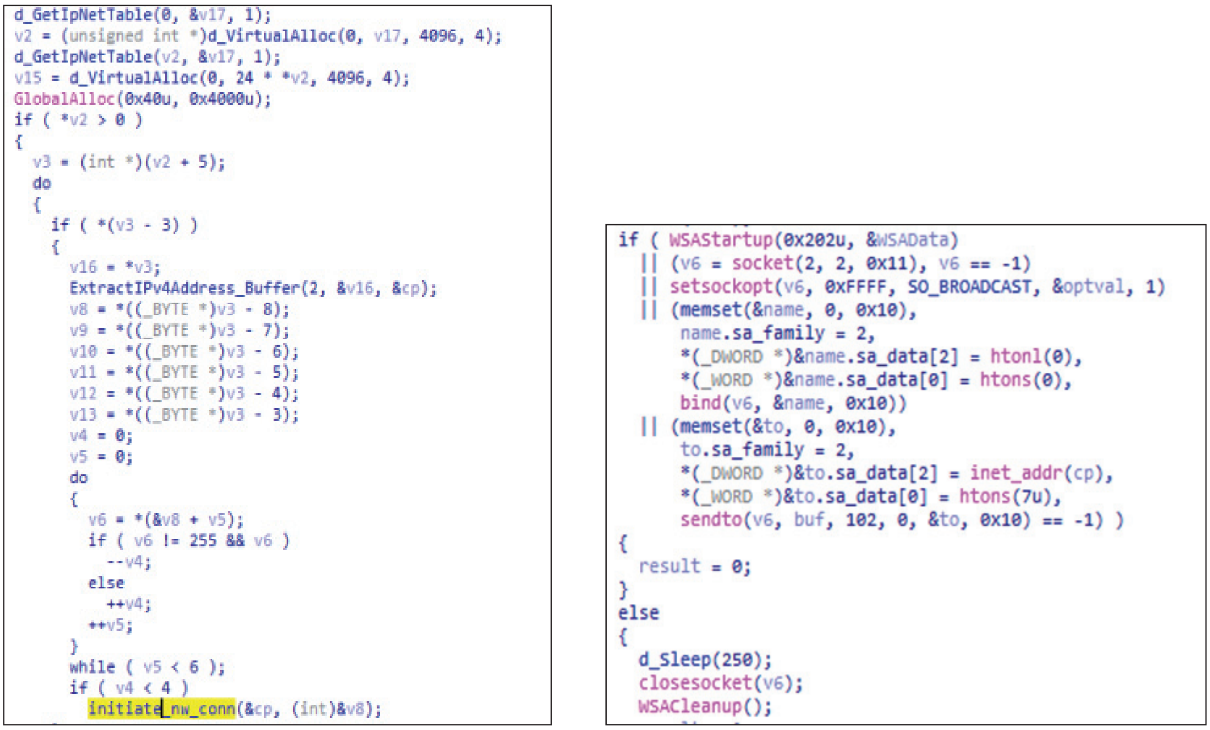
This technique can be monitored and detected via:

- Keeping watch on cross-process interaction (OpenProcess, CreateRemoteThread, ReadProcessMemory, WriteProcessMemory) via host-based monitoring tools like *Sysmon*.
- Monitoring changes to the *KeePass* config file from non-*KeePass* processes.
- PowerShell and WMI events to *KeePass* modules.

WAKE-UP, I WANNA INFECT YOU!

Wakeup-on-LAN (WoL) is a networking standard that allows a computer to be turned on or awakened by a network message. The message is usually sent to the target computer by a program executed on a device connected to the same local area network (LAN). Since this message is sent over a data link or OSI-2 layer, it is susceptible to abuse by anyone on the LAN.

In fact, we saw WoL being used last year by the Ryuk ransomware to encrypt even sleeping machines, thus increasing the reach and impact of the attack. Our preliminary analysis showed that infecting more systems via WoL helped Ryuk raise more money in the form of ransom payments. We believe that more ransomware and malware will adopt this technique in coming years.



```
d_GetIpNetTable(0, &v17, 1);
v2 = (unsigned int *)d_VirtualAlloc(0, v17, 4096, 4);
d_GetIpNetTable(v2, &v17, 1);
v15 = d_VirtualAlloc(0, 24 * *v2, 4096, 4);
GlobalAlloc(0x40u, 0x4000u);
if ( *v2 > 0 )
{
  v3 = (int *)(v2 + 5);
  do
  {
    if ( *(v3 - 3) )
    {
      v16 = *v3;
      ExtractIPv4Address_Buffer(2, &v16, &cp);
      v8 = *((_BYTE *)v3 - 8);
      v9 = *((_BYTE *)v3 - 7);
      v10 = *((_BYTE *)v3 - 6);
      v11 = *((_BYTE *)v3 - 5);
      v12 = *((_BYTE *)v3 - 4);
      v13 = *((_BYTE *)v3 - 3);
      v4 = 0;
      v5 = 0;
      do
      {
        v6 = *(&v8 + v5);
        if ( v6 != 255 && v6 )
          --v4;
        else
          ++v4;
        ++v5;
      }
      while ( v5 < 6 );
      if ( v4 < 4 )
        initiate_nw_conn(&cp, (int)&v8);
```

Extracting ARP table of system.

```
if ( WSAStartup(0x202u, &WSAData)
  || (v6 = socket(2, 2, 0x11), v6 == -1)
  || setsockopt(v6, 0xFFFF, SO_BROADCAST, &optval, 1)
  || (memset(&name, 0, 0x10),
      name.sa_family = 2,
      *(_DWORD *)&name.sa_data[2] = htonl(0),
      *(_WORD *)&name.sa_data[0] = htons(0),
      bind(v6, &name, 0x10))
  || (memset(&to, 0, 0x10),
      to.sa_family = 2,
      *(_DWORD *)&to.sa_data[2] = inet_addr(cp),
      *(_WORD *)&to.sa_data[0] = htons(7u),
      sendto(v6, buf, 102, 0, &to, 0x10) == -1) )
{
  result = 0;
}
else
{
  d_Sleep(250);
  closesocket(v6);
  WSACleanup();
```

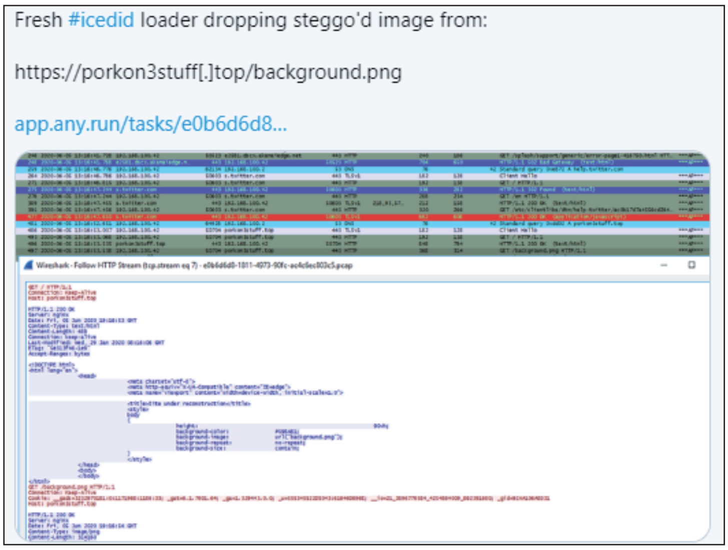
Magic packet for WoL implemented by Ryuk.

Figure 6: Code implemented by Ryuk.

A deep dive implementation of this technique in Ryuk can be seen in [5].

WILL STEGANOGRAPHY MAKE A COMEBACK?

Steganography is an old technique in which malicious code is hidden in images and other carrier files. In the past, cybercriminals have used this technique successfully to compromise machines just by getting users to visit a website where the image is hosted or simply by sending that image via email and luring the user into opening it.



Fresh #icedid loader dropping steggo'd image from:

https://porkon3stuff[.]top/background.png

app.any.run/tasks/e0b6d6d8...

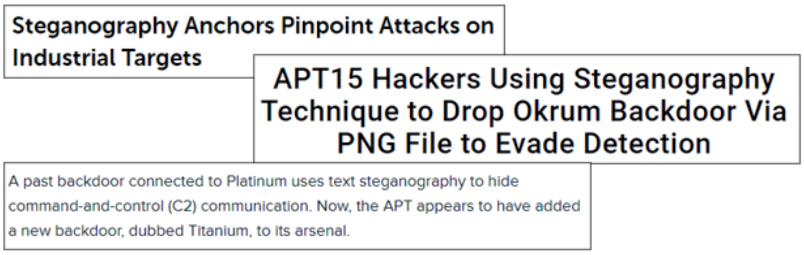
Wireshark · Follow HTTP Stream (tcp.stream eq 7) · e0b6d6d0-1011-4073-90fc-ac4c6ec003c5.pcap

Figure 7: Recent in-the-wild use of steganography technique.

We believe the steganography technique has already started its comeback. Over the past few years, there has been a notable increase in in-the-wild malware campaigns using steganography and similar tricks to embed malicious code in pictures and other carrier files. The following are some notable in-the-wild examples of the use of this technique (source: [6]):

- AdGholas – hides malicious JavaScript in image, text, and HTML files
- Cerber – embeds malicious code in image files
- DNSChanger – uses PNG LSBs to hide malware AES encryption key
- Stegano – PNG formatted banner ads containing malicious code
- Stegoloadr – this malware uses both steganography and cryptography to conceal an encrypted URL to deliver later stage payloads
- Sundown – white PNG files are used to conceal exploit code or exfiltrate user data
- SyncCrypt – ransomware that hides part of its core code in image files
- TeslaCrypt – HTML comment tags in an HTTP 404 error page contain C2 server commands
- Vawtrak – hides a URL in the LSBs of favicons to download a malicious payload
- VeryMal – malware targets *macOS* users with malicious JavaScript embedded in white bar
- Zbot – appends data to the end of a JPEG file containing hidden data
- ZeroT – Chinese malware that uses steganography to hide malware in an image of Britney Spears.

Perhaps the more worrying trend is the apparent use of steganography in targeted attacks. The technique is gradually becoming a part of the arsenal of some major cyber attack groups.



Steganography Anchors Pinpoint Attacks on Industrial Targets

APT15 Hackers Using Steganography Technique to Drop Okrum Backdoor Via PNG File to Evade Detection

A past backdoor connected to Platinum uses text steganography to hide command-and-control (C2) communication. Now, the APT appears to have added a new backdoor, dubbed Titanium, to its arsenal.

Figure 8: Articles showing steganography use by APTs.

Given the prevalence of image-based advertisements and the popularity of image sharing on social websites, we expect the use of this technique in malware to increase.

CONCLUSION

The techniques highlighted in this paper are relatively simple to implement and are also used for legitimate purposes. This means both that attackers do not have to work very hard to hide these attacks, and that it is much harder for security products to detect them. As a result, attackers have started using these techniques and we expect them to become more prevalent in times to come.

The intent of this paper is to bring these techniques to defenders' attention so that we all can work on proactively blocking attacks that use them.

Based on what we know and what we've gleaned from others' publications, and through industry sharing, the amount of information collected on such techniques is so large that it has not been possible to include all the details that relate them to threat groups. Our analysis of the sophisticated techniques will continue in several directions and we will continue to publish the details of such techniques in the coming years.

REFERENCES

[1] Bianco, D. The Pyramid of Pain. http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[2] https://github.com/HarmJ0y/KeeThief/.

[3] https://twitter.com/harmj0y.

[4] Schroeder, W. A Case Study in Attacking KeePass. https://www.slideshare.net/harmj0y/a-case-study-in-attacking-keepass.

[5] A Deep Dive Into Wakeup On Lan (WoL) Implementation of Ryuk. Quick Heal Blog. February 2020. https://blogs.quickheal.com/deep-dive-wakeup-lan-wol-implementation-ryuk/.

[6] Hiding Code Inside Images: How Malware Uses Steganography. SentinelOne. July 2019. https://www.sentinelone.com/blog/hiding-code-inside-images-malware-steganography/.