

# **Absolute Backdoor Revisited**

*Vitaliy Kamlyuk, Sergey Belov, Anibal Sacco*

*June 2014*

# Table of Contents

- [Introduction](#)
- [The Mystery](#)
- [Prior Research](#)
- [Motherboard-level Dropper](#)
  - [BIOS Option ROM](#)
- [Persistence Activation/Deactivation](#)
- [Absolute Computrace Execution Procedure](#)
  - [Stage 1: BIOS module](#)
  - [Stage 2: autochk.exe](#)
  - [Stage 3: rpcnetp.exe](#)
  - [Stage 4: rpcnet.exe](#)
- [Subject of Research](#)
  - [Rpcnetp.exe \(Small Agent\)](#)
  - [Rpcnet.exe \(Main Agent\)](#)
- [Network Protocol Description](#)
- [Remote Attacks](#)
- [Risks of Local Attacks](#)
- [Detection of Computrace Activity](#)
- [Solving The Mystery](#)
- [Threat Mitigation](#)
- [Conclusions](#)
- [Acknowledgements](#)
- [References](#)
- [Appendix A: Indicators of Computrace Agent Activity](#)
- [Appendix B: Related File Hashes](#)

## Introduction

Modern computer systems that are widely used by individual consumers as well as large corporations have a number of pre-installed software that is shipped by an OEM manufacturer or a regional reseller to promote certain services and products. It might be difficult for an ordinary user to understand all the risks of such "extra-packages" existing on the system. While most of these products can be permanently removed or disabled by the user or an IT administrator, some types of product are designed to remain on the system even after professional system cleanup or total disk drive replacement. One such type of software is anti-theft technologies that are widely used on modern laptops, i.e., Absolute Computrace. While the general idea behind anti-theft technology is good, improper implementation can render it useless as well as harmful, or even extremely dangerous. We believe that companies producing anti-theft technologies must consider the security of their products extremely seriously. Please note, that this paper does not include exploit code in order to prevent massive exploitation. Our objective is to raise awareness of IT professionals about a hidden threat, explain how it works and help protecting computer systems and networks.

## The Mystery

Our research started with a real-life incident involving one of our colleagues. He observed repeated system process crashes on one of his personal laptops. The crash generated an event log record and a memory dump that was immediately analyzed. A quick check then led to a full research cycle which eventually resulted in this report.

The failure was related to instability in modules named `identprv.dll` and `wceprv.dll` that were loaded in the address space of one of the system service host processes (`svchost.exe`). A quick analysis of the file information revealed that these modules were created by Absolute Software and are part of the Absolute Computrace software. While Absolute Software is a legitimate company and information about Computrace product is available on the company's [official website](#), the owner of the system claimed he had never installed Absolute Computrace and didn't even know the software was present on his computer. It could be assumed that the software was pre-installed by an OEM manufacturer or reseller company, but according to an Absolute Software [whitepaper](#) this should be done by users or their IT service. Unless you have a private IT service or your PC vendor took care of you, someone else has full access and control over your computer.

This single incident could have been dismissed if it wasn't for the fact that we discovered more personal computers belonging to our researchers, as well as some enterprise computers, with the same signs of Computrace working on them without authorization. From a minor hindrance the situation quickly turned into a major incident, and we decided to carry out an in-depth analysis.

## Prior Research

One of the most significant contributions previously made on this subject is authored by Alfredo Ortega and Anibal Sacco of Core Security Technologies. In their whitepaper "Deactivate the Rootkit: Attacks on BIOS anti-theft technologies" they described the general mechanisms behind anti-theft products such as Absolute Computrace.

Prior research has shown a significant risk coming from anti-theft software embedded in BIOS ROMs or firmware. It demonstrated that these modules are vulnerable to local attacks, such as those requiring physical access or the ability to run code at local system. Alfredo Ortega and Anibal Sacco demonstrated a tool that can be used to change encrypted registry settings of the Absolute Computrace Agent so that it redirects to another control server.

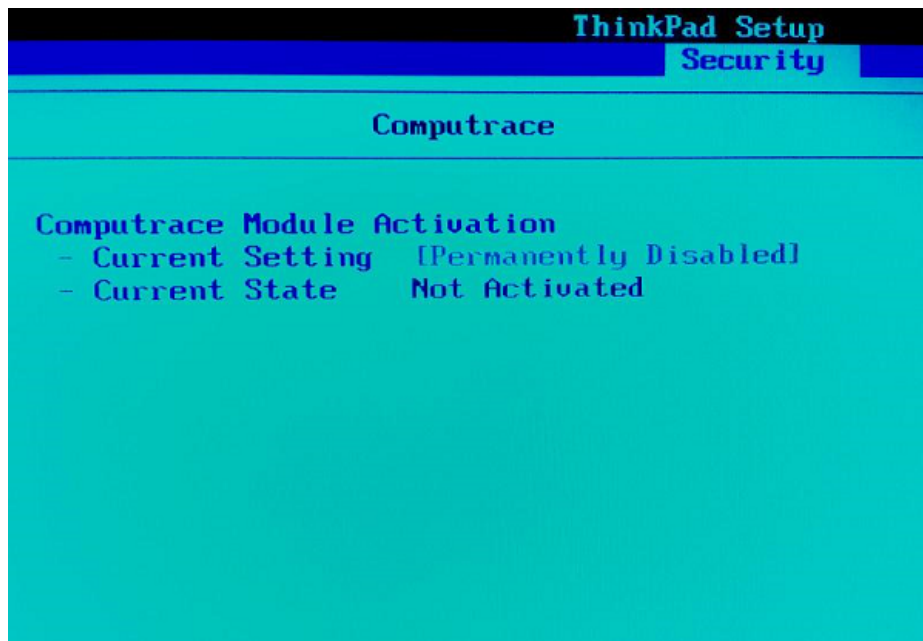
In addition, we found a [blogpost](#) authored by Bradley Susser created in August 2012. The blog mentions a vulnerability in the authentication system of LoJack (Computrace) software. However, the post didn't have enough proof to back up the claim, so we decided to embark on our own extended analysis.

## Motherboard-level Dropper

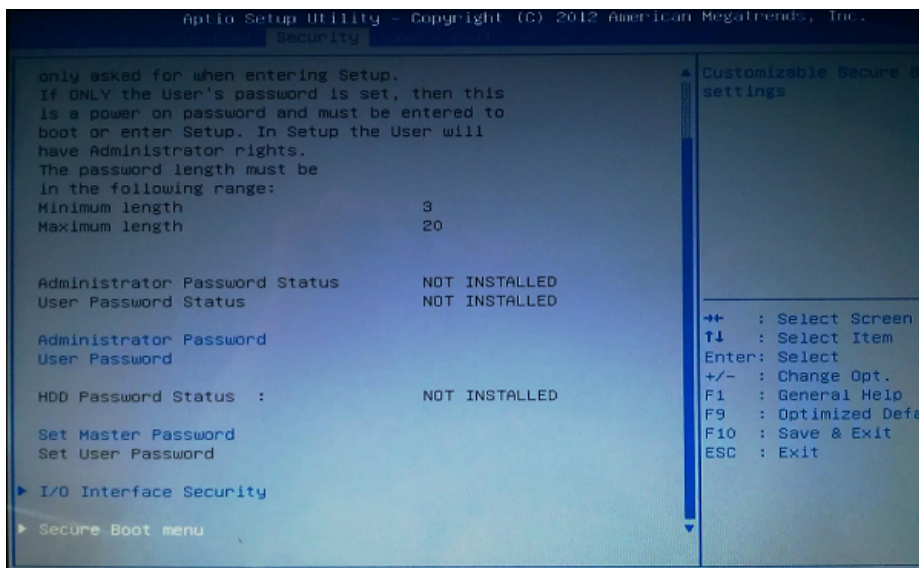
Computrace Agent is a Windows application that has two variants: a small agent and a full-size agent. The Small Agent is a piece of code that is of minimal possible size and maximum extensibility. This module is embedded into BIOS PCI Option ROM or UEFI firmware. According to the [US patent 20060272020](#) by Absolute Software, where it is called a mini CDA (Communications Driver Agent), it was designed to check if the full-function agent is installed and functioning on the system, and if not, load the full function CDA across the Internet from the server.

According to the patent, the persistence module resides in BIOS Option ROM:





**BIOS Setup Computrace settings on Lenovo Thinkpad X1**



**Computrace related settings are not visible in BIOS Setup of ASUS X102BA**

It seems that the BIOS Setup Utility developer decides whether to include the feature to enable/disable the Computrace module in BIOS Setup. There are no policies that force the developer to implement this feature. This creates a serious obstacle for ordinary users in disabling Computrace.

## Persistence Activation/Deactivation

Computrace relies on OS level API for agent persistence activation/deactivation. These procedures depend on the target vendor. For instance, in case of Toshiba products it is a direct access to CMOS registers via I/O ports 0x70/0x71. For modern UEFI systems without legacy BIOS support Get/SetFirmwareEnvironmentVar Windows API is used. If the activator cannot trigger BIOS/UEFI persistence (in case of some legacy systems) MBR modification is used instead. However, the most common activation mechanism relies on software SMIs (System Management Interrupts) that are generated when a CPU writes a magic value to specific I/O port. We have observed ports 0xB2 or 0xB0 to be used as port numbers, while values depend on data from SMBIOS tables.

```
Handle 0x0021, DMI type 12, 5 bytes
System Configuration Options
Option 1: DSN:      MT4AD1GZH0A6S5
Option 2: DSN:8E98A36E058D
Option 3: DSN:D850E63A89E8
Option 4: SMI:00B2CA
```

#### Part of dmidecode output with SMI port values

The “magic value” is set into EAX register that is used as command or signature that identifies SMI handler destination. EBX register contains another magic value that works as a “password” during activation. The password used for activation must be the same during deactivation. On the systems we analyzed this was a single call working as a flip-flop switch which enables/disables persistence and doesn't report success or failure.

```
push    ebx
mov     eax, 544241CAh
sub     esp, 18h
mov     ebx, 12345678h
mov     edx, 0B2h
out     dx, al          ; Advanced Power Management Control Port
                        ; generates SMI interrupt if APMC_EN bit is set
mov     [esp+1Ch+var_18], eax
mov     [esp+1Ch+var_1C], offset aResultX ; "result = %x\n"
call    printk
add     esp, 18h
pop     ebx
retn
```

#### Flip-flop activator sample

However, some Dell, IBM and Lenovo systems use 3 different commands for Activate/Deactivate/GetStatus operations. This fact might introduce a possibility of brute force attack with limited password length of 32bits. An attacker is able to instantly see status after each password check attempt. This is yet a theoretical claim and was not proven by practical results due to limits of research resources.

Also, we have observed changes of “OEM Strings” values which happened on each activation/deactivation SMI call. The meaning of these strings may become a subject for future research. We believe that a password and current status might be encoded here by SMI handler. The effect of erasing these strings is unclear.

```
Handle 0x0020, DMI type 11, 5 bytes
OEM Strings
String 1: voIHKSB3UVm0R
String 2: N1bTA2-Di8CG0
String 3: 5nbewuF6GBX2S
```

#### Part of dmidecode output

## Absolute Computrace Execution Procedure

Absolute Computrace product consists of several components, including BIOS/UEFI dropper, first and second stage Windows agents and their components. We divided operation of the software into 4 stages.

### Stage 1: BIOS module

The first stage (after main BIOS initialization) is to execute modules from Option ROM. At this point the Computrace code searches for available disk drives and analyzes the partition table. If FAT/FAT32/NTFS partitions are found, it locates the Windows installation path and autochk.exe application. Next, it creates a backup of system default autochk.exe code parts and overwrites them with its own code. These parts are saved as autochk.exe.bak file on FAT or autochk.exe:BAK NTFS ADS stream on NTFS filesystem. This can be used as an indicator of Computrace activity at stage one.

On some systems where the Computrace module is not part of the BIOS or it cannot be activated, a different approach is used. On such systems the Computrace activation code modifies the MBR of the hard drive and takes control of the PC at the earliest stage of the system boot. Apparently this approach is not as persistent as a BIOS-based dropper.

## Stage 2: autochk.exe

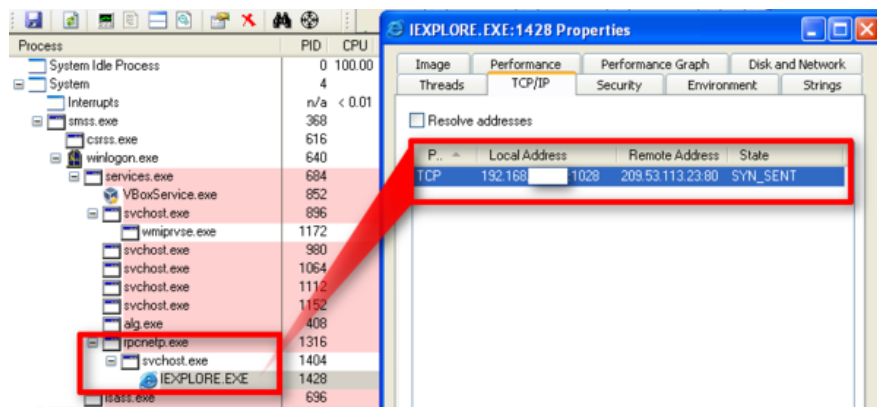
At this stage a modified autochk.exe starts and has full access to the local file system as well as system registry via Windows NT Native API calls. Its main purpose is to drop the local file rpcnetp.exe and change the local system registry to create a new system service called rpcnetp. The original autochk.exe code is then restored.

## Stage 3: rpcnetp.exe

This module is also known as the small Computrace Agent or mini CDA (Communication Driver Agent). It's approximately 17 KB in size and is written in C language.

It is started as a Windows service; however, its operation is not limited to being a system service. This Windows PE executable copies itself to another file with a .DLL extension, modifies PE header flags accordingly to change the Windows PE EXE file to a Windows PE DLL and loads it in memory. After that, rpcnetp.exe creates a child process "svchost.exe" in a suspended state and injects a freshly created rpcnetp.dll into its memory. When a DLL injection is successful and the svchost.exe process is resumed, the latter creates its own child process "iexplore.exe" started with the environment and rights of the locally logged-in user. A new iexplore.exe is started in a suspended state as well, and it receives an injection of the same rpcnetp.dll.

When iexplore.exe is resumed, it may connect to the Absolute Command & Control (C&C) server to get commands and download additional modules to execute.



**rpcnetp.exe started two extra processes to initiate a connection with the Absolute C&C server**

This technique is widely used in malicious software and was one of the reasons for a close interest in the modules. In fact, according to our experience, no other legitimate software uses techniques like this. Also, the software uses a time delay of about one minute. We assume that this delay is used to let the system find and connect to a Wi-Fi network after starting. But this is also used as a trick in many malicious applications to prevent malware detection which relies on emulators or sandboxes.

Why are there so many processes to accomplish the simple task of downloading an update? One possible answer is intentional obfuscation of the whole process to protect against reverse engineering. It is more difficult to analyze code that is running as a part of three different processes and has two variants: a DLL and an EXE file. It is known that the entry point procedure of EXE and DLL files differ in terms of the number of parameters passed at the start function. While the DLL and EXE variants start from the same executable entry point, all potential problems are dealt with quite well, so the code doesn't crash or cause any instability here.

In addition, we have seen certain anti-disassembling and anti-debugging tricks. After a successful start, rpcnetp.exe removes a registry service entry. This entry will be recreated with the next system start if rpcnetp.exe failed to connect to the C&C server.

We believe the reason it runs in the iexplore.exe process context is to guarantee the availability of the Internet. The svchost.exe process running with Local System rights starts iexplore.exe in the context of a locally logged-in user. Therefore, if the user has Internet access via a proxy server, it will be automatically used when the agent connects to the C&C server. But why keep an extra svchost.exe process? This is apparently because of the limitations of iexplore.exe rights. So far, the svchost.exe process running with Local System privileges simply exploits iexplore.exe with user rights to pass the data to and from the server. This is implemented via a number of CreateRemoteThread, WriteProcessMemory, ReadProcessMemory system API calls. Actually, the rpcnetp.dll module spawns several threads per request to the C&C server. Due to frequent thread creation and termination, the whole system works

rather slowly. We have recorded and analyzed a communication between the agent and C&C server. It's noteworthy that the agent spawned 1355 new threads in svchost.exe and 452 in iexplore.exe to download less than 150KB of data. The network communications took about four minutes on a high-speed Internet link. During this time the agent issued 596 POST requests. More details about the Small Agent network protocol can be found later in this report.

After all, the main purpose of rpcnetp.exe is to download and start a fully functional remote access tool. It communicates with a C&C server, relying on the built-in capabilities of the Small Agent to obtain some extra executables. The first executable that is sent to the agent is the file **wceprv.dll**, which is used to provide data encryption. Soon after saving wceprv.dll in the System32 directory, the Small Agent loads it in memory and switches conversation with the C&C to a more secure encrypted form. After that the Small Agent downloads extra files such as **identprv.dll**, **Upgrd.exe** and **NTAgent.exe** (later renamed to rpcnet.exe). Then it starts Upgrd.exe which is a single-run tool that handles an upgrade procedure: stopping and removing the current rpcnetp service and registering and starting a new service for rpcnet.exe ("Remote Procedure Call (RPC) Net").

## Stage 4: rpcnet.exe

Similarly to rpcnetp.exe named "Small Agent", this executable is referred by the name "Main Agent". When the rpcnet service starts successfully, it attempts to connect to the C&C server right away. The procedure is very similar to rpcnetp service: it spawns child svchost.exe which creates iexplore.exe under the local user account. Like rpcnetp it creates many threads in these processes during communication with the C&C server. This service looks for configuration in several places on the system: registry key, reserved space on a hard drive and in its own body. The configuration states what server the agent should connect to. Surprisingly, it connects back to the same server and port as the previous rpcnetp service as well as uses similar protocol. We have briefly analyzed this service and found that it was designed to provide extensible remote access to the machine running it. Comparing to the rpcnetp.exe it has extra security measures to protect against simple attacks.

## Subject of Research

We would like to note that we have worked with the most recent available modules of Absolute Computrace that we managed to find. Generally speaking there are two modules which were briefly analyzed.

### Rpcnetp.exe (Small Agent)

This module was extracted from a newly purchased laptop. The laptop was purchased in January 2014 and had one of the top configurations of its class. The executable had the following link date:

Field	Value
Count of sections	4
Symbol table	00000000[00000000]
Size of optional header	00E0
Linker version	10.00
Image version	0.00
Entry point	00002B15
Size of init data	00000A00
Size of image	00008000
Base of code	00001000
Image base	00400000
Section alignment	00001000
Stack	00100000/00001000
Checksum	00000000
Machine	Intel386 Mon Jun 18 01:45:02 2012
Magic optional header	010B
OS version	4.00
Subsystem version	4.00
Size of code	00003600
Size of unit data	00000000
Size of header	00000400
Base of data	00005000
Subsystem	GUI
File alignment	00000200
Heap	00100000/00001000
Number of dirs	16

Link date: 18 Jun 2012

Size: 17'408

MD5: 9e0e75b75b8625ad8b782d3fea62f02c

SHA1: 6117364a9edfbbcc0c850b194363aaa82d01ec8

### Rpcnet.exe (Main Agent)

This executable was extracted from a system that communicated to Absolute control server on 30th May 2014. The server pushed this executable as a replacement for rpcnetp.exe.



```

rpcnet.exe  ↓FRO ----- PE .00400000 | Hiew 8.00 <c>SEN
00400000: 4D 5A 04 00-01 00 00 00-04 00 01 00-FF FF 06 6A
00400010: 00 CB 00 00-00 00 00 00-40 00 00 00-00 00 00 00
00400020: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00

Count of sections          5
Symbol table 00000000[00000000]
Size of optional header   00E0
Linker version            10.00
Image version              0.00
Entry point                0000475F
Size of init data         00001A00
Size of image              00012000
Base of code               00001000
Image base                 00400000
Section alignment         00001000
Stack 00100000/00001000
Checksum 00000000
Overlay 0000E000[000030A0/12448/12,156 Kbl]

Machine Intel1386
Mon Feb 18 00:52:33 2013
Magic optional header 010B
OS version 4.00
Subsystem version 4.00
Size of code 0000C200
Size of uninit data 00000000
Size of header 00000400
Base of data 0000E000
Subsystem GUI
File alignment 00000200
Heap 00100000/00001000
Number of dirs 16

00400150: 00 C2 00 00-00 04 00 00-00 00 00 00-00 00 00 00
00400160: 00 00 00 00-20 00 00 60-2E 64 61 74-61 00 00 00
      .data

```

Link date: 18 Feb 2013  
 Size: 69'792  
 MD5: 675c575444aafd56b4e8a99ef8a570cd  
 SHA1: 80c7c2ae72d108cfe3b897bb23efb74865c9e5ae

## Network Protocol Description

Soon after startup, rpcnetp.exe (the Small Agent) attempts to establish a TCP connection with the C&C server. It may connect via an IP address hard-coded in its body, set in the registry or stored on a hidden location on the hard disk. If direct communication by IP fails, it may try to resolve a domain name (typically search.namequery.com) and use a new IP instead.

Small Agent is used to download and run additional executables. However, we haven't found any specific functionality that was designed for downloading and running additional modules. Nevertheless, the extensibility of its protocol allows it to do absolutely anything including downloading and running extra modules. The [US20060272020](#) patent has an interesting paragraph regarding the rpcnetp.exe communication protocol: *Deploying the Persistence Agent successfully in BIOS, for example, makes heavy use of an extensibility designed into the communications protocol. Without this extensibility the Agent would be larger and require frequent updates to add or change functionality. Such updates are neither practical nor economical, since the BIOS is programmed into the flash EEPROM of the platform and special tools (most often requiring user interaction) must be used to update the BIOS. Also, intensive testing is performed by the OEM on the BIOS since its integrity is critical to the operation of the computer.*

Having read that, it becomes clear that we should not expect to find any classic implementation of an update mechanism. According to the patent, the Small Agent supports "A method to read and write the Agent's memory space". Basically, this is the core mechanism of running arbitrary code on the remote computer, sufficient to accomplish a download task.

Here is how communication between the Small Agent and a C&C looks in Wireshark network sniffer:

```

POST / HTTP/1.1
TagId: 0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)
Host: search.namequery.com
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
Content-Type: image/jpeg
Content-Length: 17
Connection: Keep-Alive
TagId: 1342271559

~.....Gp.P...~
POST / HTTP/1.1
TagId: 1342271559
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)
Host: search.namequery.com
Content-Length: 15
Connection: Keep-Alive
Cache-Control: no-cache

~Gp.P..p.V....~HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
Content-Type: image/jpeg
Content-Length: 17
Connection: Keep-Alive
TagId: 1342271559

~.....Gp.P...~
POST / HTTP/1.1
TagId: 1342271559
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)

```

#### Part of network communication between Small Agent and C&C server

The whole communication consists of a series of POST requests and HTTP responses. The first request sent by the Small Agent has no payload; it's an empty POST request. The server replies with a special HTTP header, Tag Id, that will be used until the end of conversation with the agent. Each HTTP response and subsequent POST request includes short binary data which forms a packet to process. HTTP is used in a very simple mode just as a carrier of the agent packets. The binary packets are crafted using Computrace's basic communication protocol. While we see that the agent initiates an HTTP session and sends the first HTTP request to the server, the direction of real communication is opposite to that. The server responses are treated as request to the client and the client responds to these requests in the data added to the following HTTP POST request.

You can find the structure of such packets below:

Server Read Packet:



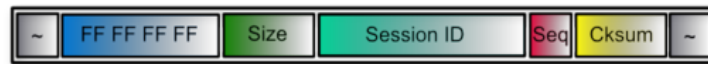
Server Write Packet:



All packets start and end with a special byte "~" (0x7E). This byte is used as a packet border indicator (packet separator). The following 4-byte field contains an **Address** of memory to work with. If the packet is of read-type, then the Small Agent will read memory starting from this address. The number of bytes to read is specified in the following **Size** field (2-bytes long). Each packet is appended with special 3 bytes: 1 byte for **Seq** value and 2 bytes for **Cksum**. **Seq** is a special 8-bit value used as a sequence number that is incremented by server and client according to their sequence algorithm. **Cksum** is a 2-byte value having a short custom hash of all the fields after packet separator and before the **Cksum**. If **Seq** or **Cksum** values are not what Small Agent expects, then current request is disposed and the last response is resent again. A **Seq** number corresponding to the last response should indicate to the C&C server that the agent has received a corrupted or altered packet and the server should retransmit the request.

The first packet of the server is special and is used for a basic handshake with the client. It looks like this:

Server Handshake Packet:



The **Address** field is set to hexadecimal 0xFFFFFFFF, **Size** is set to 0x0004 and a unique **Session ID** is chosen by the server. The handshake packet is very important as the client responds with address of control structure that can be used to extend the protocol features (i.e. call system API function).

The value of **Session ID** should be used by the client in all responses to the server. Client response has a fixed format:

Client Response Packet:



Like in the server packets, the response packet must always start and end with a packet separator. The first 4 bytes after that are set to a fixed **Session ID** value defined by the server in the Handshake Packet. Next, the 2-byte field is the size of **Response Data** which is following that value. After that **Seq** and **Cksum** fields are used as with the server packets.

An additional byte modification rule (escaping) may be applied if any of the fields between the packet separators contain a byte with hex-code 0x7E (which matches the packet separator). In this case the 0x7E byte is transformed into a sequence of two bytes 0x7D 0x5E, which would increase the packet size and affect the checksum. However, interpretation of the packet and calculation of the checksum is only accomplished after unescaping the packet. If the 0x7D byte is met in the packet before escaping, it must also be escaped and 0x7D is replaced with a sequence of 0x7D 0x5D bytes.

This completes the protocol according to what we have observed in real communications. The protocol provides two basic primitives:

1. **Read** operation
2. **Write** operation

In addition to that, the Handshake Packet provides the basic address of the Session object in the memory of the Small Agent. This might be sufficient to execute arbitrary code. However, on systems with DEP and ASLR enabled some extra steps may significantly ease the process of running code and make the process smoother and more stable. That is why there is extra processing implemented in the Small Agent. Upon receiving and writing data to a defined memory location it checks a special field in the Session object which defines a built-in basic command to execute. If the C&C server changes the value of this field, the agent may run a special command with parameters. The following commands are implemented in the module we analyzed:

- Get handle of a module in memory (calls GetModuleHandleA)
- Get address of an exported procedure (calls GetProcAddress)
- Reserve memory on heap
- Free memory on heap
- Execute chain of commands from memory location
- Call a function with specified memory address and parameters

This adds extra flexibility and allows an engineer to precisely allocate memory, transfer data in it and execute any extra code if required.

The Main Agent protocol is similar to the Small Agent protocol except few differences. The Main Agent uses the same server and port as the Small Agent. The protocol design doesn't define a special field for protocol version. So how does the server know what version of the agent is running on the client side. The developers used a trick to hint the server that it may speak a modified version (v2) of the agent's protocol. The server sends handshake request twice. While the Small Agent replies with same response, the Main Agent starts responding with modified response packets which begin with double Session ID values as illustrated below.

Client Response Packet (v2):



This double SessionID fields indicate that the client is running rpcnet.exe and not rpcnetp.exe. To better understand the protocol its worth to take a look at example of client-server communication (client packets start with "c", server packets start with "s"):

1. c
2. s 7e ff ff ff ff 04 00 e5 de 00 70 08 96 e8 7e
3. c 7e e5 de 00 70 04 00 c0 fe 88 00 09 a9 f0 7e



13. c - Yeah.. sorry, I didn't mean... Lets do that!
14. s - Good, then here is the new buffer address for input requests: 0x001b3cac.
15. c - Acknowledged.
16. s - Use same buffer size 0x0578 for output processing as well.
17. c - Of course! I do what you say.
18. s - Remind your output processing structure address.
19. c - That's easy: 0x0088fd28
20. s - Time to check if you are lying. What's the SessionId of that output processing structure?
21. c - Well.. I guess you know, it's 0x7000dee5
22. s - Alright, you're doing good. Then use it in new output processing.
23. c - I am happy to do that for you!
24. s - And for input processing too.
25. c - Absolutely!
26. s - Ok, new communication structure is ready at 0x001b423a
27. c - Sure, I got it.
- ...

The Sequence Number byte algorithm is easy to understand. It follows the rule:

1. Server increments 4 high-order bits as an integer between 0 and 7.
2. Client increments 4 low-order bits as an integer between 8 and f.

There are exceptions when it comes to transferring large chunks of data that are placed in multiple packets but this is insignificant to current research.

Checksum algorithm makes a 2-byte hash of all bytes from separator to the Checksum field. The checksum in the conversation above (see line 2) was 96 e8 which is interpreted as a WORD value 0x96e8 (note that high order byte goes first!). This value is a hash of the following data: ff ff ff 04 00 e5 de 00 70 08

A sample checksum function (cksum) implementation in Python can be as following:

```
def cksum_update(cksum,c):
    ncs = cksum
    ncs = ((ncs/256)^ord(c))*256 + ncs%256
    for i in range(0,8):
        c = ncs/32768
        ncs = (ncs*2)%65536
        if c:
            ncs = ncs ^ 0x1021
    return ncs
def cksum(data):
    cs = 0x0000
    for i in range(len(data)):
        cs = cksum_update(cs, data[i])
    return cs
```

## Remote Attacks

There are two classes of remote attacks against Absolute Computrace. The first class of attacks is directed at the Small Agent module which is executed during Stage 3. It's important to note that Small Agent runs for a limited time starting from initial installation of the module and ending when the system is connected to the Internet and module is successfully updated. Apparently an attacker has a single chance to attack Small Agent unless he has chance to reset the dropper mechanism which requires remote or local access to the machine and makes such attack almost useless. The second class of attacks is directed at the main agent module that is pushed as a replacement for the Small Agent. Current paper describes both classes of attacks.

The protocol used by the Small Agent (rpcnetp.exe) provides the basic feature of remote code execution. At the early stage of communication the protocol doesn't use encryption or authorization of the remote server, which creates numerous opportunities for remote attacks in a hostile network environment. Although encryption seems to be added to the protocol at some later stages of communication, an attacker may utilize the basic unencrypted protocol to successfully hijack the system remotely. A typical attack on a local area network would be to redirect all traffic from a computer running Small Agent to the attacker's host, i.e. by using ARP-poisoning. Another possibility is

to use a DNS service attack to trick the agent into connecting to a fake C&C server. We believe there are more ways to accomplish such attacks, though this is beyond the scope of current research.

The algorithm of attack includes the following essential remote commands:

1. Allocate heap memory and increase communication buffer size
2. Upload a data block with DLL and export function names, as well as API parameters
3. Load DLLs and get module handles
4. Get API procedure addresses
5. Call a sequence of API functions with parameters to take over the system

This attack can be fully automated and we have implemented a proof-of-concept code for this. The following CVE ID was reserved by MITRE for this vulnerability: **CVE-2014-4665**.

The protocol used by the Main Agent (rpcnet.exe) is very similar, however the agent implements some protection against fake control servers. First, it implements a protection in function invocation. The change was introduced in procedure that processes extended requests to call function by pointer. The following screenshot shows code from the Small Agent:

```
and     edx, 0Fh
cmp     edx, 8           ; switch 9 cases
ja     short loc_403A88 ; jumptable 00403A31 default case
jmp     ds:off_403ABD[edx*4] ; switch jump
; -----
loc_403A38:
; CODE XREF: rpc_command_exec+3D↑j
; DATA XREF: .text:off_403ABD↓o
shr     ecx, 4           ; jumptable 00403A31 cases 1,2
push   dword ptr [eax+4] ; dwCmdParam1
push   ecx              ; nDataLen
push   edx              ; cCmd
call   API_call
jmp     short loc_403A8E
```

Below is the same function which is modified in the Main Agent:

```
and     eax, 0Fh
cmp     eax, 0Bh        ; switch 12 cases
ja     loc_40AE25        ; jumptable 0040AD4B default case
jmp     ds:off_40AE5F[eax*4] ; switch jump
; -----
loc_40AD52:
; CODE XREF: rpc_command_exec+38↑j
; DATA XREF: .text:off_40AE5F↓o
; jumptable 0040AD4B cases 1,2
push   esi
call   check_cmdbuf_params
test   eax, eax
jnz   short loc_40AD64

loc_40AD5C:
; CODE XREF: rpc_command_exec+8F↓j
or     dword ptr [ebx], 0FFFFFFFh
jmp     loc_40AE2B
; -----
loc_40AD64:
; CODE XREF: rpc_command_exec+47↑j
movzx  eax, byte ptr [esi]
push   dword ptr [esi+4] ; a4
mov     ecx, eax
shr     ecx, 4           ; dwNumParams
push   ecx
and     eax, 0Fh
push   eax              ; a2
call   api_call
push   eax
push   esi
mov     [ebx], eax
call   sub_40AF6E
jmp     loc_40AE2B
; -----
```

There is an extra procedure (called `check_cmdbuf_params` in our listing) that allows to call only functions from the safe (allowed) list. This list can be extended by the server but this process requires to pass some additional checks.

Although, there might be more vulnerabilities in this procedure (i.e. replay attack) we have not aimed to find all of them and stopped with the first obvious bug.

```

; signed int __stdcall check_cmdbuf_params(char *lpCmdBuf)
check_cmdbuf_params proc near          ; CODE XREF: rpc_command_exec+40↓p

dwSafetyDisabled_2= dword ptr -0Ch
pAPIFunc          = dword ptr -8
i                 = dword ptr -4
pCmdBuf          = dword ptr 8

        push    ebp
        mov     ebp, esp
        sub     esp, 0Ch
        push    ebx
        mov     ebx, [ebp+pCmdBuf]
        movzx  eax, byte ptr [ebx]
        mov     ecx, [ebx+4]
        shr     eax, 4
        mov     eax, [ecx+eax*4]
        mov     [ebp+pAPIFunc], eax
        mov     eax, bSafetyDisabled
        push   esi
        xor     esi, esi
        mov     [ebp+dwSafetyDisabled_2], eax
        mov     eax, plistAllowedModules
        mov     [ebp+i], esi
        cmp     eax, esi
        jnz    short loc_40AC77
        xor     eax, eax
        jmp    short loc_40ACE8

```

There is a boolean value that we called bSafetyDisabled. Once it is set to non-zero value it allows to call any function pointer even out of the list of allowed functions. There might be a number of ways to set that variable to non-zero value. The method we used was based on the packet number 10 from the sample communication session shown previously. It includes "**Please put the result of this operation here: ...**". So, to change the boolean value we can ask the client to allocate memory and put the resulting pointer to a fixed location of bSafetyDisabled variable. Of course, an attacker has to do some extra steps such as locating current module image base and calculating the right offset of the variable, but that's easy to achieve by remotely examining the stack nearby the control structure. This attack can be fully automated and we have implemented a proof-of-concept code for this. The following CVE ID was reserved by MITRE for this vulnerability: **CVE-2014-4666**.

## Risks of Local Attacks

Unfortunately, the rpcnetp.exe module has no digital signature inside and no file information data. This makes it not only problematic for a system administrator to decide if it is a legitimate application but also makes it difficult to validate the integrity of the module's code. We believe this is a serious flaw in current module implementation. In addition, if an attacker sets a system attribute for the modified file, it will never be replaced by a legitimate copy of rpcnetp.exe from BIOS.

As it was explained earlier in this report (see description of the small agent), both rpcnet.exe and rpcnetp.exe rely on an specific configuration block to obtain the C&C server address before attempting to establish a TCP connection with it.

In regards to the registry configuration block, it is only protected by a simple encryption method (a sequence of XOR operations against fixed 8bit values). Previous research demonstrated that, it is quite simple to decode/modify/re-encode this block in order to get the main agent redirected to a custom C&C server.

Likewise, we demonstrate that it is possible to slightly modify the configuration block contained (at offset 0x3c00) in rpcnetp.exe. In this case, the configuration block is only protected by a XOR operation against a single byte value (0xb5). This modification will force the agent to establish a connection to a custom C&C server offering full control over the system. This is achieved with a single executable of Computrace on the system.

3C00h:	04 02 00 00 80 1E 04 01 00 40 00 1F 04 00 00 00	....ε....θ.....
3C10h:	00 10 0A F4 F4 85 F8 84 EC 85 85 85 85 1D 02 00	...ôô...ø,,i.....
3C20h:	00 46 06 00 00 00 00 00 00 47 06 00 00 00 00 00	.F.....G.....
3C30h:	00 48 22 B5 E5 64 80 C4 A2 C6 D0 D4 C7 D6 DD 9B	.H"μάδεΆοΑΒΔΌζΌΥ>
3C40h:	DB D4 D8 D0 C4 C0 D0 C7 CC 9B D6 DA D8 B5 B5 B5	ÛÔØΔΆΔÇÏ >ΌÚμμμ
3C50h:	B5 B5 B5 B5 B5 0A 02 07 10 06 06 00 00 00 00 00	μμμμμ.....
3C60h:	00 07 06 00 00 00 00 00 00 0F 06 B6 69 CE 05 05	.....ϖiî..
3C70h:	96 08 06 19 99 08 08 12 12 0B 02 93 03 14 04 39	-...™.....`...9
3C80h:	00 80 00 20 04 00 00 00 00 15 04 00 00 00 00 19	.ε. ....
3C90h:	1B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
3CA0h:	00 00 00 00 00 00 00 00 00 00 00 00 1A 01 00 1B	.....
3CB0h:	06 00 00 00 00 00 00 2D 01 B8 2D 01 B8 55 02 00	.....-.-.U..
3CC0h:	00 33 01 B8 2B 04 F4 E1 F1 E1 28 03 00 00 00 01	.3. ,+.ôáñá{.....
3CD0h:	2C 01 33 01 B8 2B 04 F4 E1 F1 E1 28 03 00 00 00	,.3. ,+.ôáñá{....
3CE0h:	01 1B 01 00 00 00 00 00 00 00 00 00 00 00 00	.....
3CF0h:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

(Before decoding)

3C00h:	B1 B7 B5 B5 35 AB B1 B4 B5 F5 B5 AA B1 B5 B5 B5	±·μ5«±'μδμ²±μμμ
3C10h:	B5 A5 BF 41 41 30 4D 31 59 30 30 30 30 A8 B7 B5	μΨζΑΑΟΜ1ΥΟΟΟΟ"·μ
3C20h:	B5 F3 B3 B5 B5 B5 B5 B5 B5 F2 B3 B5 B5 B5 B5 B5	μó²μμμμμμμó²μμμμμ
3C30h:	B5 FD 97 00 50 D1 35 71 17 73 65 61 72 63 68 2E	μý-.PÑ5q,search.
3C40h:	6E 61 6D 65 71 75 65 72 79 2E 63 6F 6D 00 00 00	namequery.com...
3C50h:	00 00 00 00 00 BF B7 B2 A5 B3 B3 B5 B5 B5 B5 B5	.....ζ·²Ψ²μμμμμ
3C60h:	B5 B2 B3 B5 B5 B5 B5 B5 B5 BA B3 03 DC 7B B0 B0	μ²²μμμμμμμ°°·Û{°°
3C70h:	23 BD B3 AC 2C BD BD A7 A7 BE B7 26 B6 A1 B1 8C	#²²-,²²SS²·ε¶;±ε
3C80h:	B5 35 B5 95 B1 B5 B5 B5 B5 A0 B1 B5 B5 B5 B5 AC	μ5μ²±μμμμμ ±μμμμμ-
3C90h:	AE B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5	@μμμμμμμμμμμμμμμμ
3CA0h:	B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 AF B4 B5 AE	μμμμμμμμμμμμμ'μ@
3CB0h:	B3 B5 B5 B5 B5 B5 B5 98 B4 0D 98 B4 0D E0 B7 B5	²μμμμμμμ"'. "'.à·μ
3CC0h:	B5 86 B4 0D 9E B1 41 54 44 54 9D B6 B5 B5 B5 B4	μ†'.ž±ATDT.¶μμμ'
3CD0h:	99 B4 86 B4 0D 9E B1 41 54 44 54 9D B6 B5 B5 B5	™'†'.ž±ATDT.¶μμμ
3CE0h:	B4 AE B4 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5	'@'μμμμμμμμμμμμμμ
3CF0h:	B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5 B5	μμμμμμμμμμμμμμμμ

(After decoding)

This becomes a major risk when we have in consideration that the Computrace agent makes use of common malware techniques to function but, at the same time, it is established as a legitimate product. It is whitelisted by most anti-malware vendors which, because of the lack of an unambiguous identification method like a digital signature, fall in a file hash-dependent mechanism to identify it. This way it leaves an open door for malicious attackers to use it as a fully functional "legal" connect-back mechanism to be deployed after a successful intrusion.

## Detection of Computrace Activity

Computrace Agent doesn't use any Windows rootkit to hide the processes, which is why the most easiest way to detect agent's activity is to check the list of running processes. If you have **rpcnetp.exe** or **rpcnet.exe** processes running on your system, it means Computrace is there.

However, if Computrace is used by a potential attacker checking the process names might not be enough. It's worth of checking local filesystem for presence of Computrace agent modules. To quickly check your system you can use free Yara tool<sup>3</sup> and the following rules file:

```
rule ComputraceAgent
{
  meta:
```

<sup>3</sup> <http://plusvic.github.io/yara/>



```

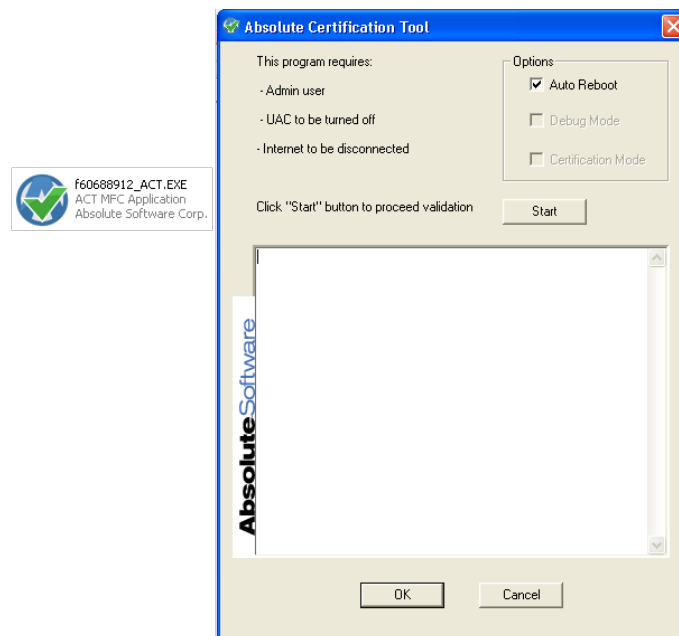
description = "Absolute Computrace Agent Executable"
thread_level = 3
in_the_wild = true
strings:
  $a = {D1 E0 F5 8B 4D 0C 83 D1 00 8B EC FF 33 83 C3 04}
  $mz = {4d 5a}
  $b1 = {72 70 63 6E 65 74 70 2E 65 78 65 00 72 70 63 6E 65 74 70 00}
  $b2 = {54 61 67 49 64 00}
condition:
  ($mz at 0 ) and ($a or ($b1 and $b2))
}

```

## Solving The Mystery

During our research we kept wondering who or what activated the BIOS dropper in our systems. We have applied computer forensics methods to understand when the modules were initially started and found that this date matched the day of the first system start. It means that the laptop was purchased with Computrace agent being on the filesystem or with pre-activated persistence mechanism. It may also indicate that persistence mechanism could have been activated by the manufacturer.

Why would a manufacturer need to activate persistence? We are uncertain, however we found a possible explanation to that. During our preparation for a live demo we have purchased a brand new laptop in a local retail store. It had no signs of Computrace agent running on it, which was good. Next we searched the raw disk image of the laptop hdd for signs of Absolute modules and found that the disk had files in unallocated space of the filesystem. When we applied file recovery to the unallocated space we managed to extract a number of files including some documents, emails, scripts and executables which seemed to be unintentionally left by the manufacturer. One of the recovered executables had internal name ACT which stands for Absolute Certification Tool. That looked like a tool developed by Absolute Software that laptop manufacturers could use to test current BIOS/UEFI for Computrace compatibility.



Link Date: 10 Aug 2012

Size: 421'888

MD5: 6c6e3b3ff374839122f36192d7bf2aa8

SHA1: 0477e94a4e17c133b86d0870a3a42da9c8216f37

The tool requires administrative privileges and UAC disabled for the time of testing. It has 4 stages

1. Activate BIOS/UEFI dropper
2. Reboot and check that rpcnetp.exe IS running
3. Deactivate BIOS/UEFI dropper

#### 4. Reboot and check that rpcnetp.exe IS NOT running

We disabled UAC and started the tool under local administrator account. It successfully passed stage 1 and 2 and then suddenly got stuck at stage 3 outputting an error with not technical details. So far, stage 3 couldn't be passed for current configuration which means that the BIOS/UEFI dropper remained activated.

Although there was no evidence of intentional secret activation of Computrace modules on the computers we analyzed, we believe that the number of computers with Computrace activated may be surprisingly high due to such failures to deactivation of the BIOS/UEFI dropper during automated testing stage.

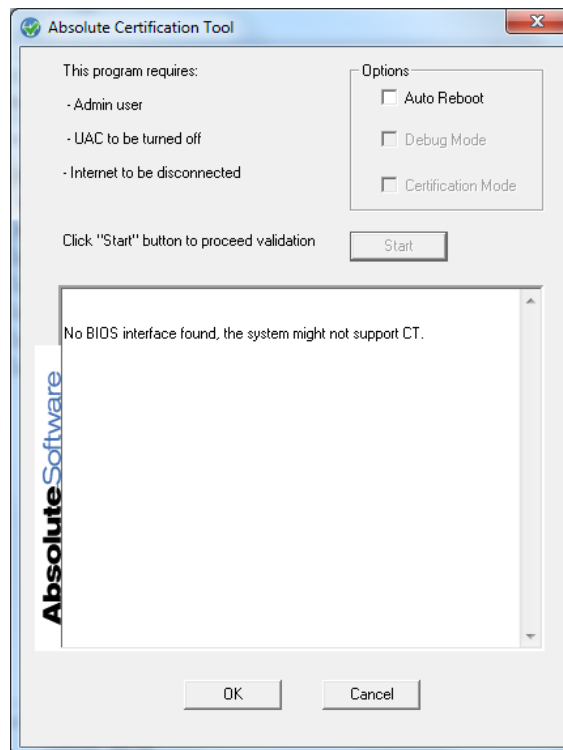
## Threat Mitigation

The ACT tool contains four Windows applications inside which are executed to activate/deactivate persistence mechanism.

- instac32.exe - MBR persistence, for some Dell systems without BIOS/UEFI support
- instb32.exe - IBM and Lenovo systems
- instd32.exe - Dell
- instt32.exe - Toshiba
- instm32.exe - others

There is a couple of drivers for 32bit and 64bit Windows inside each module(except MBR version). These drivers have low level primitives, such as in/out operations, memory mapping from physical to userland address space, low address memory allocation. High level logics is placed in exe file. These executables are started by ACT with the following command line option "-i12345678" for activation, and "-d12345678" for deactivation. Where "12345678" is used as an installation password. It is converted to a dword integer and as a magic value during activation. It's also required for successful deactivation. We have tested it on a brand new laptop we purchased for this research. We have successfully one of these executables to activate and deactivate persistence with testing password "12345678".

Also, we have tried to deactivate the laptop from which this story began. Unfortunately, our attempts have failed.



This was unexpected. We see Computrace agent running in the OS as well as Computrace modules inside BIOS of this system. It seems that ACT password 12345678 doesn't match the password of activation on this system. We expected that all erroneously activated systems could be deactivated with the same password from ACT application, however our experiments showed that it's not so.

This is why, the most reliable way to disable unauthorized or erroneously activated persistence is to kill Computrace

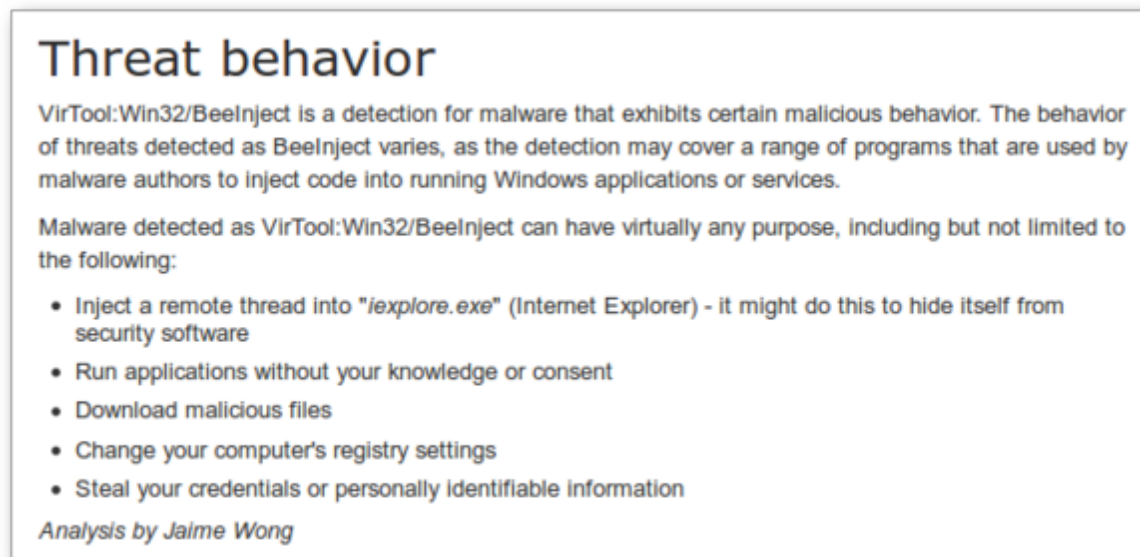
processes, make zero-sized files rpcnetp.exe, rpcnetp.dll in System32 and SysWOW64, and set System file attributes. You can reboot and remove the rest of Computrace files (see Appendix A for full file list).

Additionally we would like to recommend contacting Absolute Software technical support to help you permanently deactivate persistence if you are legitimate owner of the system and have not subscribed to Absolute Software services.

## Conclusions

When we first found and analyzed Computrace we mistakenly thought it was malicious software, because it used so many tricks that are popular in actual malware. It has specific anti-debugging and anti-reverse engineering techniques, injects into the memory of other processes, establishes secret communication, patches system files on disk (autochk.exe), keeps configuration files encrypted, and finally drops a Windows executable directly from BIOS/UEFI.

Such aggressive behavior by Computrace Agent was the reason it was detected as malware in the past. According to some [reports on the Internet](#), Computrace was detected by Microsoft as VirTool:Win32/BeelInject. Here is how Microsoft [describes](#) this generic threat name:



**Threat behavior**

VirTool:Win32/BeelInject is a detection for malware that exhibits certain malicious behavior. The behavior of threats detected as BeelInject varies, as the detection may cover a range of programs that are used by malware authors to inject code into running Windows applications or services.

Malware detected as VirTool:Win32/BeelInject can have virtually any purpose, including but not limited to the following:

- Inject a remote thread into "iexplore.exe" (Internet Explorer) - it might do this to hide itself from security software
- Run applications without your knowledge or consent
- Download malicious files
- Change your computer's registry settings
- Steal your credentials or personally identifiable information

*Analysis by Jaime Wong*

Nevertheless, detection of Computrace modules was later removed by Microsoft and some AV vendors. Computrace executables are currently whitelisted by most AV companies.

We believe that Computrace was designed with good intentions, but our research shows that vulnerabilities in this software can turn a useful tool into a powerful weapon for cybercriminals. We believe that such a powerful tool needs to have powerful authentication and encryption mechanisms to continue fighting the good fight. We have no reasons to think that Absolute Software or any PC manufacturers secretly activate persistence, but it's clear that if there are a lot of computers with activated Computrace agents, it is the responsibility of the manufacturers and Absolute Software to notify those users and explain how they can deactivate it if they don't want to use Absolute Software services. Otherwise, these orphaned agents will keep on running unnoticed and provide opportunities for remote exploitation.

We have contacted the vendor (Absolute Software) and notified about existing vulnerabilities. The vendor has responded and confirmed receiving technical descriptions of our analysis. While the vendor has denied existence of vulnerabilities in its products, we were promised that these "security issues" will be addressed.

## Acknowledgements

We would like to thank

1. Alfredo Ortega for previous contribution to this topic
2. Costin Raiu, Eugene Kaspersky and Kaspersky Lab for supporting this research

## References

1. <http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=publication&name=Deactiva>

- [te the Rootkit](#)
2. <http://www.absolute.com/en/partners/bios-compatibility>
  3. <http://www.absolute.com/en/resources/whitepapers/absolute-persistence-technology>
  4. <https://www.google.com/patents/US20060272020>
  5. <http://en.wikipedia.org/wiki/LoJack>
  6. [https://www.securelist.com/en/analysis/204792325/Absolute\\_Computrace\\_Revisited](https://www.securelist.com/en/analysis/204792325/Absolute_Computrace_Revisited)
  7. [http://www.securelist.com/en/blog/208216083/Absolute\\_Computrace\\_Frequently\\_Asked\\_Questions](http://www.securelist.com/en/blog/208216083/Absolute_Computrace_Frequently_Asked_Questions)

## Appendix A: Indicators of Computrace Agent Activity

1. One of the following processes is running:
  - a. rpcnet.exe
  - b. rpcnetp.exe
  - c. 32-bit svchost.exe running on 64-bit system (can't serve as complete indicator)
2. One of the following files exist on the hard drive:
  - a. %WINDIR%\System32\rpcnet.exe
  - b. %WINDIR%\System32\rpcnetp.exe
  - c. %WINDIR%\System32\wceprv.dll
  - d. %WINDIR%\System32\identprv.dll
  - e. %WINDIR%\System32\Upgrd.exe
  - f. %WINDIR%\System32\autochk.exe.bak (for FAT filesystems)
  - g. %WINDIR%\System32\autochk.exe.bak (for NTFS filesystems)
  - h. %WINDIR%\System32\wctguard.exe
  - i. %WINDIR%\System32\wctsys.exe
  - j. %WINDIR%\System32\btlcpl.cpl

Please note, than on 64bit Windows instead System32 it should be SysWOW64 subfolder of Windir
3. The system resolves one of the following domain names using DNS:
  - a. search.namequery.com
  - b. search.us.namequery.com
  - c. search64.namequery.com
  - d. bh.namequery.com
  - e. namequery.nettrace.co.za
  - f. search2.namequery.com
  - g. m229.absolute.com or any m\*.absolute.com
4. The system connects to the following IP: 209.53.113.223
5. One of the following registry keys exist:
  - a. HKLM\System\CurrentControlSet\Services\rpcnet
  - b. HKLM\System\CurrentControlSet\Services\rpcnetp

## Appendix B: Related File Hashes

Below are some of files that were discovered as **rpcnetp** or **rpcnet** service binaries:

0153ad739956b12bf710c7039186728d  
01a19f74cfb19cc61d62009bcfa59961  
076a360ee0cfc5ca2afc8468fa1ae709  
130206a40741aa57f3778bb70e593e16  
19a51da66e818f0e10973e1082c79a70  
19e67bd685019dafadfe524517dab145  
1f2d10f767c7145a8d2a3fbbf66bed7a  
27d43a7f03260ebdf81dd6515646510b  
3a1ed2730cee3ec7d6d5091be5071eaa

418f527e59508480cfc17644d8387736  
4476ccfd883c603cebbc317c6c41c971  
4a3b02ac2e1635c0a4603b32d447fbb2  
4bcf98b48bee5e7094d0cf026d4edce4  
5235a32d018b79f065c64b06bd4001be  
5515c17117a37fc808fc7a43a37128b0  
5829887d2304c08237a5f43c42931296  
5a5bb037b8e256a3304f113a187b1891  
5e071026cb4c890a3584e02af1e3daf8  
6846e002291086843463238e525c8aaa  
77f57671b08e539e3232bf95a2ac8aec  
78c696e5fd0041d8a5ce5e5e15b6f2f3  
7a7cd44a4113046869be5ab8341f759f  
8282e68524af7a46afc1bac2105c6cda  
86332af92a6a80660bb8659711378140  
8f95ce32c2596771174f7054a78f4a84  
925f2df6a96637d23c677b33a07b52c1  
961d7bbefa57d1b260db075404454955  
a9e0a97c29bd110f54beb465d8ec3e52  
aaae16f8cbd6a35c0f6b37358b3ce54  
b4c3723eb687b0e63aeea2974b8d73ba  
b7534d5ed3b01ff3a96b43b855b2a103  
bb7ef397f31c184f4089fc9bac04566f  
c1b19ad11821780b67f4c545beb270c0  
c6089ec6ae62fe264896a91d951d0c79  
cacebf514be693301c1498e216b12dbb  
cbb0d507e47d7f0ae3e5f61ea8feff08  
cde233aa0676f5307949c0a957a2f360  
cf8bcf7138cc855d885271c4ee7e8a75  
d2561d67e34ff53f99b9eaab94e98e2a  
e2e9dcce8d87608e4ba48118b296407f  
e57892858a7d3a7799eacb06783bd819  
e583977f36980125c01898f9e86c6c87  
ed9b58f56a13fbb44c30d18b9b5c44d0  
ee08ce8247ffb26416b32d8093fe0775  
eeab12e6f535ee0973b3ddb99287e06c  
ef8d08b07756edc999fbc8cfac32dc23  
f03f740fde80199731c507cdd02eb06e  
f259382b6fa22cae7a16d2d100eb29e4  
f42dbd110320b72d8ff72f191a78e5d5  
fc0ba4c9a301b653ee2c437e29ed545e