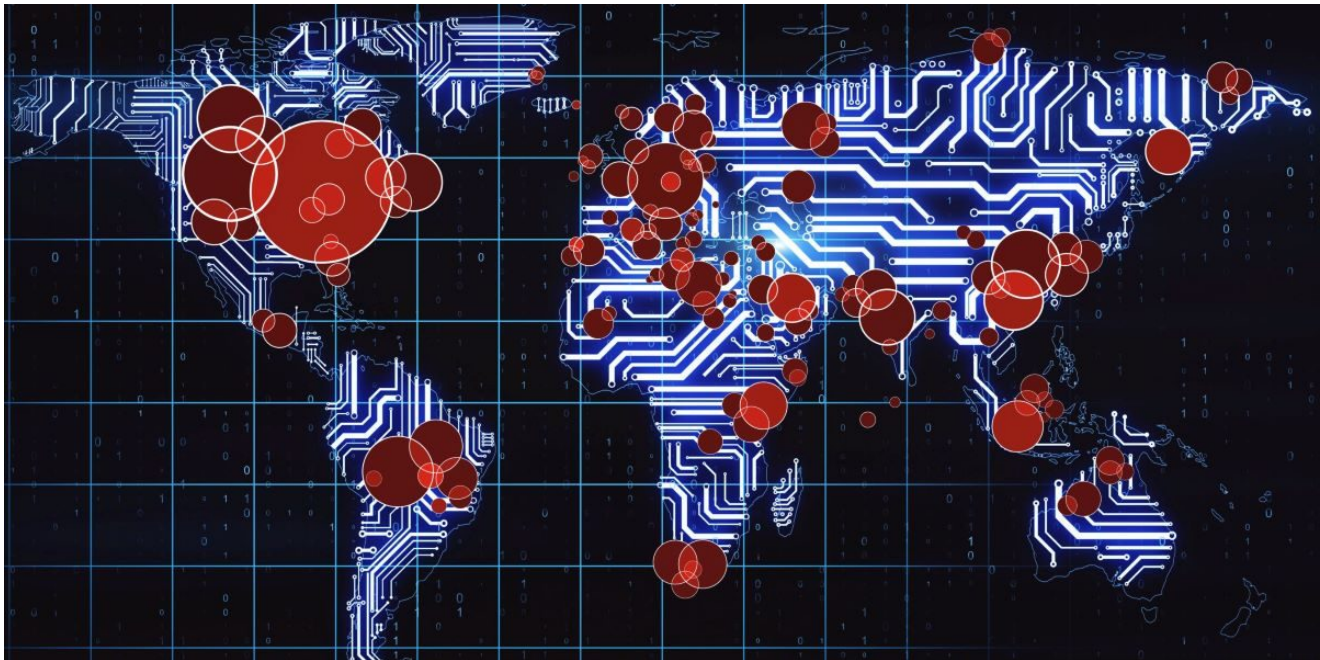



# DTrack activity targeting Europe and Latin America

SL [securelist.com/dtrack-targeting-europe-latin-america/107798/](https://securelist.com/dtrack-targeting-europe-latin-america/107798/)



Authors

- **Expert** [Konstantin Zykov](#)
-  [Jornt van der Wiel](#)

## Introduction

DTrack is a backdoor used by the Lazarus group. Initially discovered in [2019](#), the backdoor [remains in use](#) three years later. It is used by the Lazarus group against a wide variety of targets. For example, we've seen it being used in financial environments where ATMs were breached, in [attacks on a nuclear power plant](#) and also in targeted ransomware attacks. Essentially, anywhere the Lazarus group believes they can achieve some financial gain.

DTrack allows criminals to upload, download, start or delete files on the victim host. Among those downloaded and executed files already spotted in the standard DTrack toolset there is a keylogger, a screenshot maker and a module for gathering victim system information. With a toolset like this, criminals can implement lateral movement into the victims' infrastructure in order to, for example, retrieve compromising information.

As part of our crimeware reporting service, we published a new private report about recent Dtrack activity. In this public article we highlight some of the main findings shared in that report. For more information about our crimeware reporting service, please contact [crimewareintel@kaspersky.com](mailto:crimewareintel@kaspersky.com).

## So, what's new?

---

DTrack itself hasn't changed much over the course of time. Nevertheless, there are some interesting modifications that we want to highlight in this blogpost. Dtrack hides itself inside an executable that looks like a legitimate program, and there are several stages of decryption before the malware payload starts.

### First stage – implanted code

---

DTrack unpacks the malware in several stages. The second stage is stored inside the malware PE file. To get it, there are two approaches:

- offset based;
- resource based.

The idea is that DTrack retrieves the payload by reading it from an offset within the file or by reading it from a resource within the PE binary. An example of a decompiled pseudo function that retrieves the data using the offset-based approach can be found below.

```
2 void dtrack_implant(void)
3
4 {
5     code *payload_buf;
6     uint *key1;
7     uint *key2;
8     uint payload_size;
9
10    payload_size = 0;
11    payload_buf = (code *)0x0;
12    read_payload_by_offset(&payload_buf,&payload_size);
13    if ((payload_buf != (code *)0x0) && (payload_size != 0)) {
14        key1 = (uint *)read_key1();
15        key2 = (uint *)read_key2();
16        decrypt_payload_rc4((int)payload_buf, (int)payload_buf, payload_size, *key1, *key2);
17        (*payload_buf)();
18    }
19    return;
20 }
```

***Example of DTrack offset-oriented retrieval function***

After retrieving the location of the next stage and its key, the malware then decrypts the buffer (with a modified RC4 algorithm) and passes control to it. To figure out the offset of the payload, its size and decryption keys, DTrack has a special binary (we have dubbed it 'Decrypt config') structure hidden in an inconspicuous part of the PE file.

## Second stage – shellcode

The second stage payload consists of heavily obfuscated shellcode as can be seen below.

```

seg000:000053AF ; -----
seg000:000053AF
seg000:000053AF loc_53AF: ; CODE XREF: seg000:0000539C↑j
seg000:000053AF          push   eax
seg000:000053B0          jmp    loc_5518
seg000:000053B5 ; -----
seg000:000053B5 ; START OF FUNCTION CHUNK FOR sub_3DA3
seg000:000053B5 loc_53B5: ; CODE XREF: sub_3DA3:loc_80FC↓j
seg000:000053B5          xor     esi, 34D689B3h
seg000:000053BB          add     esi, 16DAF590h
seg000:000053C1          add     esi, 3A1F2103h
seg000:000053C7          xor     esi, 0EA4C9A1h
seg000:000053CD          jmp    loc_5510
seg000:000053CD ; END OF FUNCTION CHUNK FOR sub_3DA3
seg000:000053D2 ; -----
seg000:000053D2          jmp    loc_5457
seg000:000053D7 ; -----
seg000:000053D7          jmp    short loc_5448
seg000:000053D9 ; -----
seg000:000053D9 ; START OF FUNCTION CHUNK FOR sub_2501
seg000:000053D9 loc_53D9: ; CODE XREF: sub_2501+1EB8↑j
seg000:000053D9          mov     ecx, [esi+ebp]
seg000:000053DC          pop     esi
seg000:000053DD          push   ecx
seg000:000053DE          call   sub_FA3
seg000:000053E3          add     esp, 8
seg000:000053E6          push   edx
seg000:000053E7          mov     edx, 286D72D8h
seg000:000053EC          sub     edx, 8F463538h
seg000:000053F2          sub     edx, 0D8CD5BDDh
seg000:000053F8          xor     edx, 0AC2C598Ah
seg000:000053FE          sub     edx, 10C6D732h
seg000:00005404          add     edx, 0A219800h
seg000:0000540A          xor     edx, 2E7F3CBBh
seg000:00005410          xor     edx, 0B450BA58h
seg000:00005416          mov     [edx+ebp], eax
seg000:00005419          pop     edx
seg000:0000541A          jmp    loc_7B77
seg000:0000541F ; -----

```

### Heavily obfuscated second stage shellcode

The encryption method used by the second layer differs for each sample. So far, we have spotted modified versions of RC4, RC5 and RC6 algorithms. The values of the third stage payload and its decryption key are obtained by reading Decrypt config again.

One new aspect of the recent DTrack variants is that the third stage payload is not necessarily the final payload; there may be another piece of binary data consisting of a binary configuration and at least one shellcode, which in turn decrypts and executes the final payload.

### Third stage – shellcode and final binary

---

The shellcode has some quite interesting obfuscation tricks to make analysis more difficult. When started, the beginning of the key (used to decrypt the final payload) is searched for. For example, when the beginning of the key is 0xDEADBEEF, the shellcode searches for the first occurrence of 0xDEADBEEF.

```
2 void decrypt_chunk0(void)
3
4 {
5     uint *key;
6     uint payload_offset;
7
8     for (key = get_return_address(); (*key != L'\x8e05dacd' || (*(short *) (key + 2) != 0x5bf9));
9         key = (uint *) ((int) key + 1)) {
10    }
11
12     /* decrypting [payload_size][payload_entry_point_offset] */
13     dtrack_chunk_custom_decrypt((byte *) (key + 4), 8, (byte *) key);
14     payload_offset = key[5];
15     dtrack_chunk_custom_decrypt((byte *) (key + 6), key[4], (byte *) key);
16     (*(code *) (payload_offset + (int) (key + 6))) ();
17     return;
18 }
```

#### Chunk decryption routine example

Once the key is found, the shellcode uses it to decrypt the next eight bytes after the key, which form yet another configuration block with **final payload size** and its **entry point offset**. The configuration block is followed by an encrypted PE payload that starts at the **entry point offset** after decryption with the custom algorithm.

### Final payload

---

Once the final payload (a DLL) is decrypted, it is loaded using process hollowing into *explorer.exe*. In previous DTrack samples the libraries to be loaded were obfuscated strings. In more recent versions they use API hashing to load the proper libraries and functions. Another small change is that three C2 servers are used instead of six. The rest of the payload's functionality remains the same.

### Infrastructure

---

When we look at the domain names used for C2 servers, a pattern can be seen in some cases. For example, the actors combine a color with the name of an animal (e.g., pinkgoat, purplebear, salmonrabbit). Some of the peculiar names used in the DTrack infrastructure can be found below:

Domain	IP	First seen	ASN
pinkgoat.com	64.190.63.111	2022-03-03 15:34	AS47846
purewatertokyo.com	58.158.177.102	2022-05-20 16:07	AS17506
purplebear.com	52.128.23.153	2021-01-08 08:37	AS19324
salmonrabbit.com	58.158.177.102	2022-05-20 09:37	AS17506

## Victims

---

According to KSN telemetry, we have detected DTrack activity in Germany, Brazil, India, Italy, Mexico, Switzerland, Saudi Arabia, Turkey and the United States, indicating that DTrack is spreading into more parts of the world. The targeted sectors are education, chemical manufacturing, governmental research centers and policy institutes, IT service providers, utility providers and telecommunications.

## Conclusions

---

The DTrack backdoor continues to be used actively by the Lazarus group. Modifications in the way the malware is packed show that Lazarus still sees DTrack as an important asset. Despite this, Lazarus has not changed the backdoor much since 2019, when it was initially discovered. When the victimology is analyzed, it becomes clear that operations have expanded to Europe and Latin America, a trend we're seeing more and more often.

## IOCs

---

### C2 domains

[pinkgoat\[.\]com](http://pinkgoat[.]com)  
[purewatertokyo\[.\]com](http://purewatertokyo[.]com)  
[purplebear\[.\]com](http://purplebear[.]com)  
[salmonrabbit\[.\]com](http://salmonrabbit[.]com)

### MD5

[1A74C8D8B74CA2411C1D3D22373A6769](#)  
[67F4DAD1A94ED8A47283C2C0C05A7594](#)

- [APT](#)
- [Backdoor](#)

- [Lazarus](#)
- [Malware](#)
- [Malware Descriptions](#)
- [Malware Technologies](#)
- [Trojan](#)

## Authors

-  **Expert** [Konstantin Zykov](#)
-  [Jornt van der Wiel](#)

DTrack activity targeting Europe and Latin America

---

Your email address will not be published. Required fields are marked \*