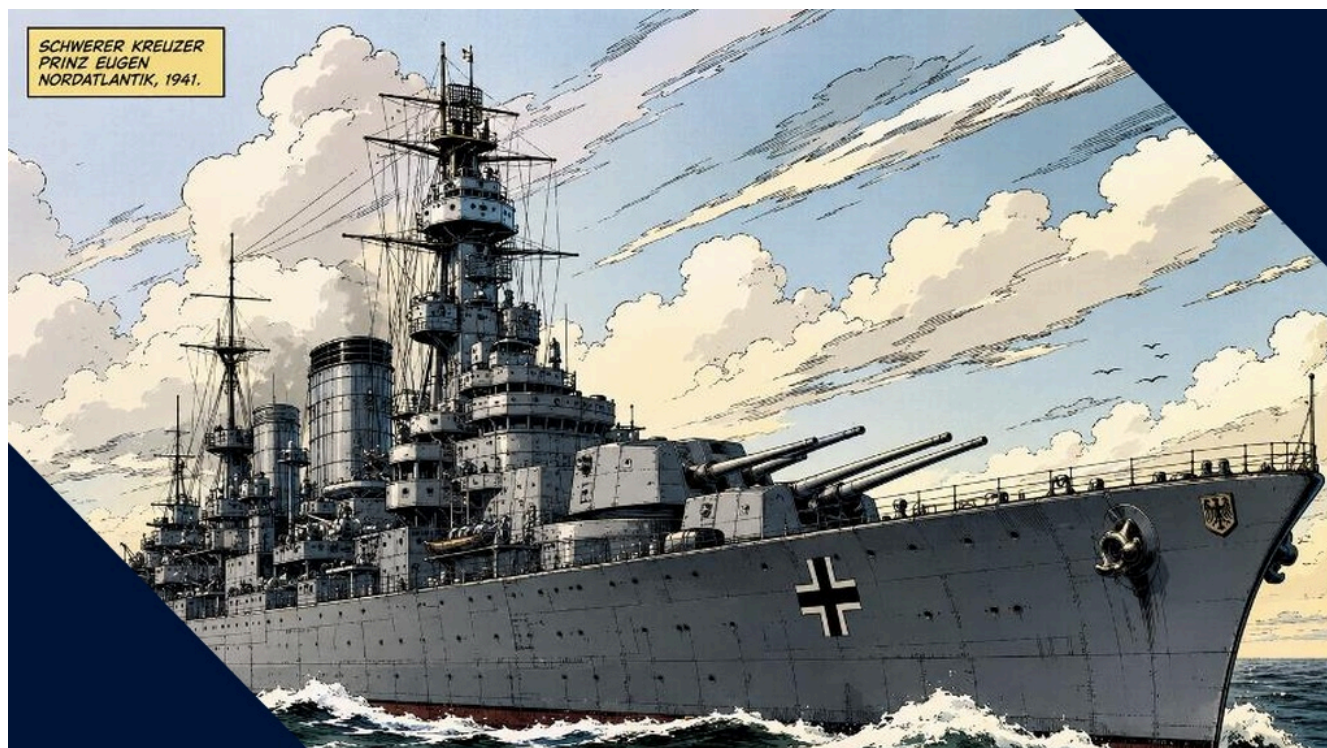


# Prinz Eugen ransomware: a deep dive into a new Go-based encryptor

 [threatdown.com/blog/prinz-eugen-ransomware-a-deep-dive-into-a-new-go-based-encryptor](https://threatdown.com/blog/prinz-eugen-ransomware-a-deep-dive-into-a-new-go-based-encryptor)

June 18, 2026



## [Ransomware](#), [Threat Intelligence](#)

Prinz Eugen is a new Go-based ransomware family that encrypts files, prioritizes fresh data, and extorts victims out-of-band, leaving a deliberate, anti-forensic footprint.

### [Andrea Bianchini](#)

On May 11, 2026, our research team investigated a customer infected with a brand-new ransomware family called Prinz Eugen.

The encryptor is freshly built, written in Go, and more technically deliberate than many first-wave ransomware samples. It performs recursive encryption, prioritizes recently modified files, uses ChaCha20-Poly1305 with integrity checks, and leaves no ransom note on disk.

The first public report related to this family is dated April 16, when a public social media post noted that a new ransomware leak portal had appeared to extort Standard Bank Group, a leading financial institution in South Africa.

We attributed the activity to the Prinz Eugen group because the encryptor appends the **.prinzeugen** extension to encrypted files.

The term “Prinz Eugen” could refer to a German heavy cruiser that served during World War II. It is the first of several German references woven through this campaign.

## Inside the encryptor

---

The following analysis traces the encryptor through the intrusion we investigated, in the order the activity unfolded: initial access, execution, the encryption process, and the anti-forensic steps the binary took before it exited.

We suspect the actor gained a foothold through compromised RDP credentials. Once inside, they downloaded the encryptor executable (**servertool.exe**) using Chrome and moved it into the user’s Music folder. The binary takes the directory to encrypt as a command-line parameter, with an optional `--delete` flag to remove the original file after encryption:

```
C:\Users\*****\Desktop>servertool.exe
usage: encrypter [--delete] <dir> [dir ...]
```

In the environment we studied, it was launched against a specific set of folders, recorded in our EDR logs:

```
servertool --delete C:\Users\<redacted>\Downloads
servertool --delete C:\Users\<redacted>\OneDrive
servertool --delete C:\Users\<redacted>
servertool --delete C:\Users
servertool --delete C:\
servertool --delete G:\My Drive
servertool --delete G:\Shared Drive
```

We analyzed the sample to clarify how it encrypts. It is written in Go, and all the key functions related to encryption sit inside a package named **scorched-earth-ausfc**.

For each directory passed on the command line, the encryptor performs a fully recursive walk with no depth limit. The code allows exclusions to be configured, but the sample we analyzed did not define any, so it is set to encrypt every file that does not already carry the **.prinzeugen** extension and is not a **.<originalname>.tmp** working file.

The encryptor takes the most recently modified files first, falling back to alphabetical order only to break ties on identical timestamps.

```
uint64_t scorched-earth-ausfc/internal/fswalk.IterFiles.func2(int64_t arg1, void* arg2, int64_t arg3 @ rax, int64_t arg4 @ rbx, void* arg5 @ r14)
1400fd4c0 char rax_1; // compare modification times
1400fd4c0 int64_t rcx_1;
1400fd4c0 rax_1 = time.Time.Equal(*(uint64_t*)(r11 + (r10 << 3) + 0x10),
1400fd4c0 *(uint64_t*)(r11 + (r10 << 3) + 0x18),
1400fd4c0 *(uint64_t*)(r11 + (rdx_1 << 3) + 0x18),
1400fd4c0 *(uint64_t*)(r11 + (rdx_1 << 3) + 0x10));
1400fd4c0
1400fd4c7 if (!rax_1) // modification times are DIFFERENT
1400fd4c7 {
1400fd52d     int64_t r9_1 = rdx[1];
1400fd52d
1400fd539     if (arg3 >= r9_1)
1400fd539     {
1400fd585         runtime.panicBounds(rcx_1);
1400fd585         /* no return */
1400fd539     }
1400fd539
1400fd53b     int64_t rdx_5 = *(uint64_t*)rdx;
1400fd53b
1400fd55a     if (arg4 >= r9_1)
1400fd55a     {
1400fd580         runtime.panicBounds(*(uint64_t*)(rdx_5 + (r10 << 3) + 0x20));
1400fd580         /* no return */
1400fd55a     }
1400fd55a
1400fd56b     *(uint64_t*)(rdx_5 + (rdx_1 << 3) + 0x20);
1400fd570     // newest modification time first
1400fd570     uint64_t rax_5 = time.Time.After(*(uint64_t*)(rdx_5 + (r10 << 3) + 0x10),
1400fd570         *(uint64_t*)(rdx_5 + (r10 << 3) + 0x18),
1400fd570         *(uint64_t*)(rdx_5 + (rdx_1 << 3) + 0x18),
1400fd570         *(uint64_t*)(rdx_5 + (rdx_1 << 3) + 0x10));
1400fd57a     return rax_5;
1400fd4c7 }
1400fd4c7
```

File iteration logic with the most recently modified files displayed first.

That ordering is what makes it dangerous. The most recently modified files are the ones most likely to be in active use (open documents, current databases, recently saved project files, fresh email archives) and the least likely to have a recent backup. Hitting them first puts the heaviest pressure on the victim to pay quickly.

## Encryption logic

Encryption is done through multiple worker goroutines, one per CPU:

```
int64_t numOfWorkers = (int64_t)runtime.numCPUStartup;
```

Each worker then calls the per-file encryption routine:

```
// spawn N worker goroutines
for (int64_t j = numOfWorkers; j > 0; j -= 1)
{
    int64_t (** rax_17)(int64_t arg1, void* arg2, int64_t arg3 @ r14,
        int512_t arg4 @ zmm0, int512_t arg5 @ zmm1, int128_t arg6 @ zmm15);
    rax_17 = runtime.newobject(rcx_11, zmm1_12, &data_140139320, arg3);
    // worker function
    *(uint64_t*)rax_17 = main.main.func1;
```

A worker function (encryption) is executed for each worker.

For a file named **document.docx**, the encryptor works in three stages. It first creates and encrypts a temporary copy, **.document.docx.prinzeugen.tmp**, using the **EncryptFileToKey**

function:

```

void** rax_11; // CREATE temp file
int64_t rcx_5;
int512_t zmm1_5;
rax_11 = os.OpenFile(0xc1,
    os.Remove(rcx_4, rdx_2, rax_7, var_130, 1, rdi_2, arg7), rax_7, var_130,
    0x180, arg7);

if (var_130)
    return var_130;

int128_t* rax_12; // Create 256KB buffer (0x40000)
int64_t rcx_6;
int512_t zmm1_6;
rax_12 = runtime.makeslice(0x40000, zmm1_5, &data_14011dfc0, 0x40000, arg7);
int128_t* rax_13; // Setup buffered writer over temp file
int512_t zmm1_7;
rax_13 = runtime.newobject(rcx_6, zmm1_6, &data_140134380, arg7);
*(uint64_t*)((char*)rax_13 + 0x18) = 0x40000;
rax_13[2] = 0x40000;
int128_t* rdx_3;
void** r12;

if ((* (uint32_t*)runtime.writeBarrier))
{
    int128_t** r11_1 = runtime.gcWriteBarrier2();
    rdx_3 = rax_12;
    *(uint64_t*)r11_1 = rdx_3;
    r12 = rax_11;
    r11_1[1] = r12;
}
else
{
    rdx_3 = rax_12;
    r12 = rax_11;
}

rax_13[1] = rdx_3;
rax_13[3] = &go:itab.*os.File,io.Writer;
*(uint64_t*)((char*)rax_13 + 0x38) = r12;
int32_t var_288 = r8;
int32_t var_284 = r9;
int32_t var_280 = r10;
char var_27c = r11;
int128_t* rdi_3 = rax_13;
char* rax_15; // ENCRYPT temp file
int64_t rcx_7;
int512_t zmm1_8;
rax_15 = scorched-earth-ausfc/internal/crypto.EncryptFileToKey(
    &go:itab.*bufio.Writer,io.Writer, zmm1_7, arg6, arg5, arg7);

```

Temporary file encryption.

It then renames that temporary file to its final form, **document.docx.prinzeugen**:

```
int64_t rax_31; // rename temp → final
uint64_t rcx_14;
int64_t rdx_12;
int512_t zmm1_11;
int128_t zmm15_2;
rax_31 = os.rename(var_68, rax_7, var_130, rbx_1, arg7);
```

Renaming the temporary file to its final version.

Where the --delete flag is set, it verifies the encrypted file is decryptable with **VerifyEncryptedWithKey** and only then removes the original:

```

if (!arg8) // f --delete flag NOT set
|   return 0; // done, original kept

var_288 = var_19d_1;
int128_t var_268;
int128_t var_278_1 = var_268;
int128_t* rsi_5 = &var_15d;
int128_t var_268_1 = var_15d;
var_248 = var_e8;
int128_t var_228;
int128_t var_238_1 = var_228;
int128_t var_218;
int128_t var_228_1 = var_218;
char rax_33;
uint64_t rcx_17;
int64_t rdx_13;
rax_33 = scorched-earth-ausfc/internal/crypto.VerifyEncryptedWithKey(&var_248,
    zmm1_11, rbx_1, arg7, zmm15_2, var_268_1, var_18d, var_248);

if (rbx_1) // verification ERROR (exception)
{
    uint64_t var_58 = rcx_17;
    // delete bad .prinzeugen
    os.Remove(rcx_17, rdx_13, var_68, rbx_1, rsi_5, rbx_1, arg7);
    return rbx_1;
}

if (rax_33) // verification PASSED
|   // DELETE original "document.docx"
|   return os.Remove(rcx_17, rdx_13, arg3, arg4, rsi_5, rbx_1, arg7);

```

Verifying and deleting the original file.

No ransom-note functionality exists in the sample. There is no code to drop a text file, HTML page, or any other written demand onto a victim's file system.

The encryptor uses ChaCha20-Poly1305 (AEAD) with a 32-byte master key, per-file random IVs, a three-stage KDF (Argon2id to SHA-256 to HKDF-SHA256), chunked encryption at 1MB per chunk, SHA-256 integrity hashing, and a custom file header. The header carries the CHV1 magic bytes, visible in an encrypted sample:

FD desktop.ini.prinzeugen

| Offset (h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text       |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|
| 00000000   | 43 | 48 | 56 | 31 | 01 | 01 | 00 | 00 | 01 | 00 | 01 | 00 | 00 | 00 | 02 | 00 | CHV1.....          |
| 00000010   | 00 | 00 | 10 | 3B | 93 | 29 | 04 | 32 | D8 | 02 | C3 | 53 | 8C | 72 | 6C | EF | ...;" ).20.ÅS@rli  |
| 00000020   | 8B | E2 | D3 | 02 | 04 | 88 | 75 | 9D | AA | 00 | 00 | 10 | 00 | 1A | 01 | 00 | <âÓ..^u.*.....     |
| 00000030   | 00 | 00 | 00 | 00 | 00 | 2A | 01 | 00 | 00 | 6D | 02 | 4D | 83 | 0E | F3 | 31 | .....*....m.Mf.ó1  |
| 00000040   | 9C | 58 | F9 | 33 | AD | 8E | D0 | ED | 9F | BC | 45 | 20 | E2 | 88 | D7 | 9C | œXù3.ŽĐiY4E â^×œ   |
| 00000050   | 5E | E0 | CE | E3 | 65 | E3 | F0 | 9B | 4F | F0 | D8 | E4 | 10 | 81 | 46 | 49 | ^âîãeãð>0ððä..FI   |
| 00000060   | 52 | 23 | 3C | 63 | 30 | E9 | 0C | 1A | 79 | 73 | 9B | 46 | 5B | 81 | 2E | BD | R#<c0é...ys>F[...¼ |
| 00000070   | 0D | 57 | 70 | F9 | 16 | 7B | 95 | 91 | 7F | 12 | A9 | 66 | CE | 4E | 55 | 71 | .Wpù.{.'...@fîNUq  |
| 00000080   | 0C | 38 | 99 | A5 | 43 | 5C | 30 | 95 | 2F | 35 | 1A | 91 | 15 | DB | 99 | E1 | .8™¥C\0•/5.'.'Ô™á  |
| 00000090   | 87 | 09 | 0E | BF | BF | 76 | 08 | D7 | F8 | B9 | 17 | 42 | 51 | D1 | 2B | 91 | ±...¿¿v.*ø³.BQÑ+^  |

An example of an encrypted file displaying the visible magic header.

Before exiting, the binary takes deliberate steps to frustrate forensic analysis and recovery. It zeroes the hardcoded encryption key and runs the garbage collector so the key does not persist in memory, then deletes itself.

```
return os.Exit(main.selfDelete(rcx_18, zmm1_16, arg3), rax_26, arg3);
```

The sample removes itself prior to termination.

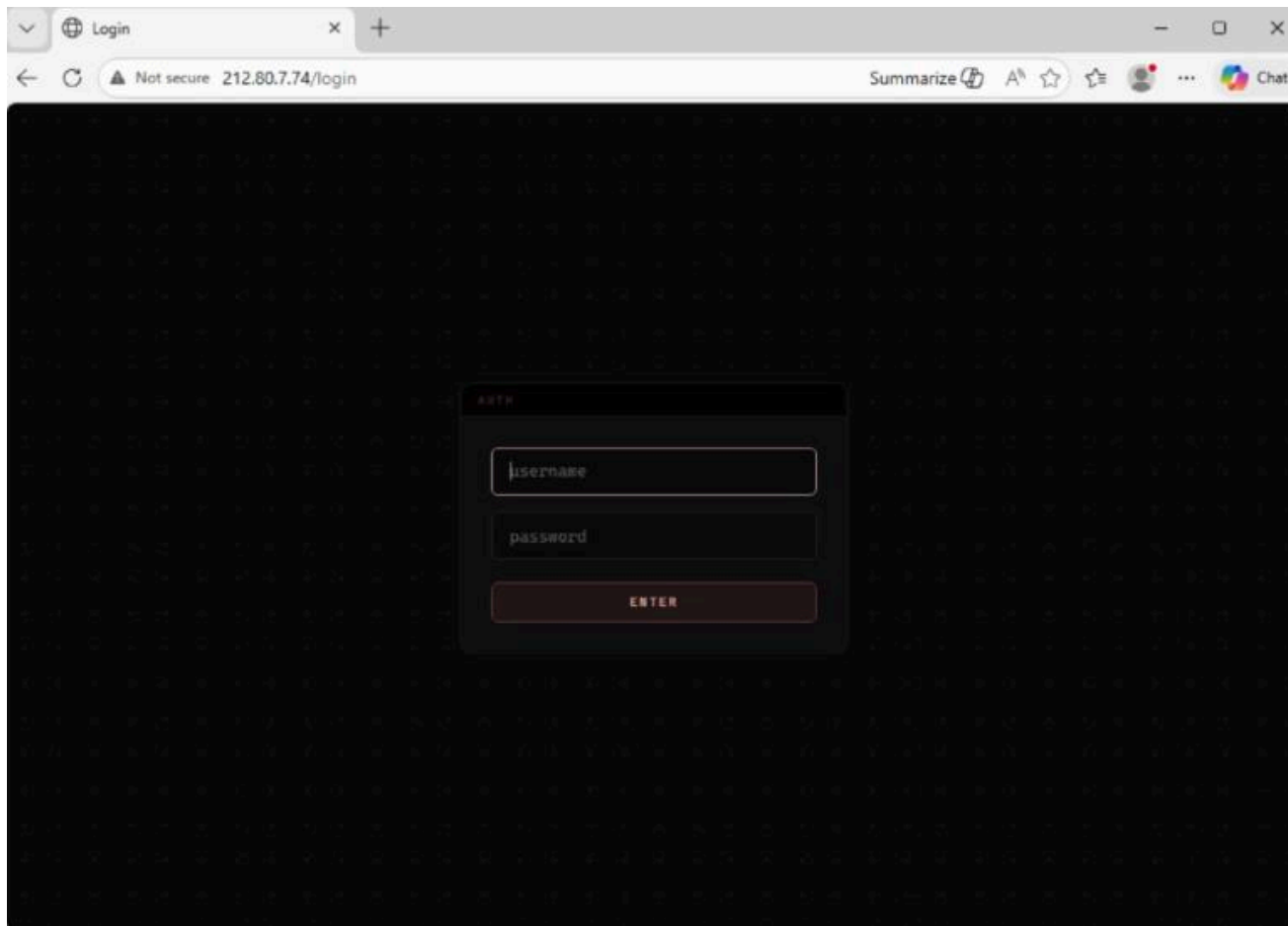
Together these reduce the forensic footprint: memory analysis will not recover the key, and the binary leaves no trace on disk after running. The self-delete uses **cmd.exe** with a ping-delay trick, which gives the parent process time to exit fully before deletion runs:

```
cmd.exe /C ping 127.0.0.1 -n 2 > nul & del /F /Q C:\Users\<redacted>\Music\servertool.œ
```

## Additional steps by the threat actor

In the environment hit by this ransomware, we also observed the actor leveraging the RemotePC RMM tool to launch PowerShell stagers and deploy additional payloads.

These payloads were downloaded from **212.80.7.74**, which, at the time of writing, was online and hosting an admin console. We suspect this IP is acting as a C2, and the additional payloads are probably remote access trojans (RATs) used for infostealing and exfiltration.



The C2 panel at the time of writing.

We observed the actor setting up an admin account as well:

```
net user admin germania /add
```

It's worth noting the password, *germania*, which means “Germany” in Latin and Italian. *Germania* is also an extortion alias that the same operator used on a crime forum months before Prinz Eugen existed, connecting this binary directly to a named, pre-existing data seller.

So, RDP was probably the entry point, while RMM and a custom admin account became persistence mechanisms.

## Infrastructure

---

The C2 address sits inside a small infrastructure cluster. DNS history reconnaissance and URL-scan history show that three domains have resolved to 212.80.7.74:

| Domain                | Observed role              | Notes                                                                                                                                                                                                                                                |
|-----------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stndrdbnk.cc          | Standard Bank<br>typosquat | Captured serving /unlocked on March 29, consistent with a victim payment or negotiation portal.<br>Registered on March 14 2026 through Porkbun, later flagged as malicious by several reputation engines, and its DNS record has since been removed. |
| g-captchafestung.sbs  | Fake-CAPTCHA lead          | Registered on May 27 through NiceNIC. The naming suggests a possible ClickFix-style paste-and-run lure, though retrieval returned only {"ok":true}, so ClickFix remains unconfirmed.                                                                 |
| festung-e.duckdns.org | Dynamic-DNS host           | Observed between May 23 and 30.                                                                                                                                                                                                                      |

Two of those domains contain “Festung,” the German word for fortress, which matches the German theme already visible in the binary (the cruiser name, the germania password, and the scorched-earth-ausfc package).

By May 30, the host had been cleaned up. Port 443 still responded, but only with a 28-byte placeholder page last modified May 23; the admin panel shown above was gone, the CAPTCHA domain returned only {"ok":true}, and the typosquat’s DNS record had been pulled.

The same pattern appears across the actor’s estate: the C2 was stripped, the typosquat DNS was removed, and their older forum profile was deleted.

They appear to dismantle their own infrastructure after each operation. (DNS history records also show unrelated co-residents on the IP, darkempire.fun in 2024-2025 and old-pidop.ru in 2024, most likely prior tenancy with no observed actor link.)

## Unmasking the operator

The strongest public attribution trail for Prinz Eugen points to an actor called ROOTBOY. Attributing the leak-site post to a named handle relies on two consistent routes:

1. South African press reporting on the Standard Bank breach names the actor as ROOTBOY, who posted the data on DarkForums.
2. Public leak-site tracking lists a single affiliate for the Prinz Eugen site: ROOTBOY.

The ROOTBOY handle also has useful historical context. The actor has posted as ROOTBOY on Exploit and DarkForums, and earlier as avtokz on XSS, linking the two accounts by a shared TOX ID while also advertising Telegram and Jabber contacts. Prior reporting connects the same actor cluster to data-sale and extortion activity that predates Prinz Eugen:

- A Vantage Finance database, advertised as avtokz on July 9, 2025.
- A US and Canada driving-school software provider in September 2025 (SQL INSERT dumps containing billing records and student PII).
- The 700Credit breach in November 2025, roughly 8.4 million US records including Social Security numbers, dates of birth and employment data, advertised on Exploit and DarkForums.

That history also reaches directly into our sample. Our analysis recorded the user setting up an account called “germania,” while in the 700Credit listing, ROOTBOY stated that the data was sold after a failed extortion attempt issued under the name GERMANIA.



The screenshot shows two forum posts by the user ROOTBOY. The top post is titled "USA | 700Credit Databreach | 8.4 Million FULLZ SSNs" and was posted on November 16, 2025. The text of the post describes a data breach at 700Credit, mentioning a failed extortion attempt by GERMANIA and listing the contents of the database: Full Name, Date of Birth, Address, Social Security Numbers, and Employment Info. The bottom post is a reply titled "[USA-CA] 5.9 Million Student Info & 10.2 Million Billing Info" dated September 18, 2025, stating that the data has not been sold yet.

**USA | 700Credit Databreach | 8.4 Million FULLZ SSNs**  
ROOTBOY posted a topic in Auctions

Hello, In late Oktober, 700Credit experienced a databreach of 8.4 Million customer records, they could not pay GERMANIA's full asking price so now you can pay for their data. В конце октября компания 700Credit пережила утечку данных 8,4 миллионов клиентов. Они не смогли заплатить полную цену, которую запросила GERMANIA, и теперь эти данные можете купить вы. The database contains: - Full Name - Date of Birth - Address - Social Security Numbers - Employment Info (Occasionally) Start: \$2500 Step: \$500 Blitz: \$5000 INSERT INTO "main"."data"...

🕒 November 16, 2025    💬 2 replies

---

**[USA-CA] 5.9 Million Student Info & 10.2 Million Billing Info**  
ROOTBOY replied to ROOTBOY's topic in Auctions

Yes, has not been sold yet.

🕒 September 18, 2025    💬 5 replies

ROOTBOY's 700Credit auction post naming GERMANIA.

A string recovered from the binary therefore matches an extortion alias that the same actor used on a forum months earlier, connecting our reverse engineering to a named, pre-existing seller. It is the single strongest link in the attribution.

On current evidence this is most likely a single person, reusing the same handle, TOX ID and German naming theme across operations, which is the very consistency that makes attribution possible.

There are two caveats worth stating here: public leak-site tracking labels ROOTBOY an “affiliate,” which can imply a multi-actor structure, and a bespoke Go encryptor implies that at

least a separate developer may be involved.

The core technical finding remains the same: Prinz Eugen is a new Go-based encryptor with deliberate file-targeting and anti-forensic behavior. The attribution context adds a second layer. It makes the absence of a ransom note, the out-of-band communication model, and the emphasis on exfiltration look intentional.

## Impact

---

The victim set is opportunistic and spread across countries and sectors, with no clear single-sector focus:

| Victim                                     | Country      | Detail                                                                                              |
|--------------------------------------------|--------------|-----------------------------------------------------------------------------------------------------|
| Standard Bank Group (+ Liberty)            | South Africa | ~1.2 TB / ~154M SQL rows exfiltrated; ~3-week dwell time; 1 BTC demand refused; staged daily leaks. |
| Transitions Pro Centre Val de Loire        | France       | Hundreds of GB exfiltrated and encrypted; posted on the leak site (May 3 2026).                     |
| 700Credit                                  | USA          | ~8.4M records incl. SSNs; sold for ~\$2,500 (pre-Prinz Eugen data sale).                            |
| Vantage Finance                            | Unspecified  | Database sold as avtokz (pre-Prinz Eugen).                                                          |
| US/Canada driving-school software provider | USA / Canada | Billing records and student PII sold (pre-Prinz Eugen).                                             |

## The extortion model

---

The available reporting and incident data point to a simple pressure model: steal the data, encrypt the environment, then use the leak site and direct contact channels to force a decision.

Reporting on the Standard Bank intrusion adds further tradecraft: a hands-on dwell time of around three weeks before exfiltration, and lateral movement through enterprise applications including SharePoint, OneDrive, Power Apps, AppDynamics, Jira, Confluence, Citrix, Remedy, and Microsoft and Oracle SQL databases.

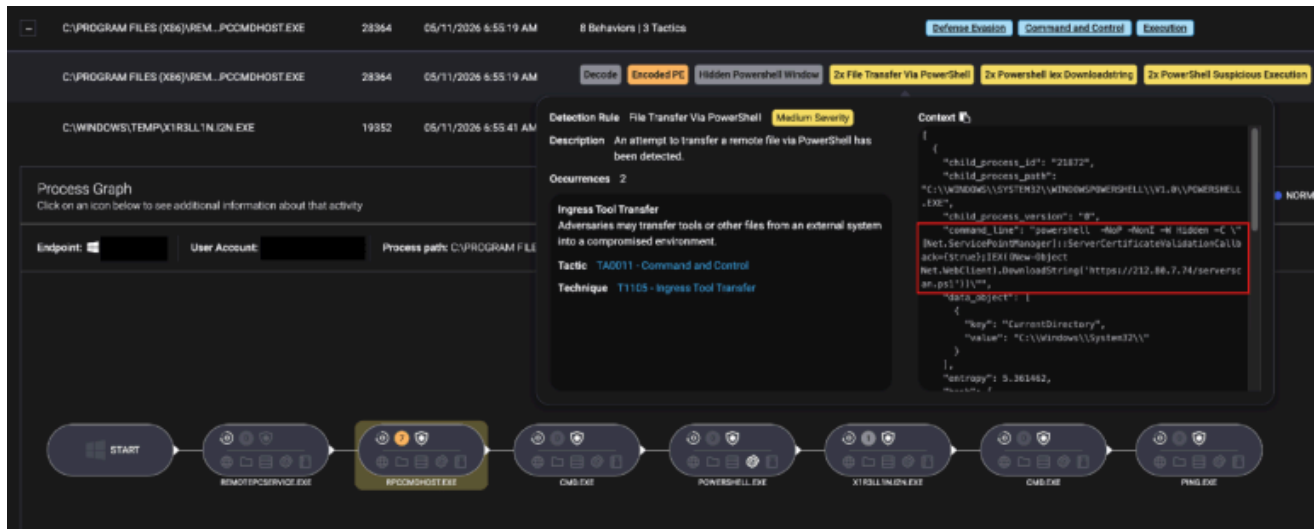
As in the incident we investigated, RemotePC was abused to launch PowerShell stagers and a manual backdoor admin account (germania) was created. The operator favors legitimate RMM tooling to blend into enterprise traffic, a Living off The Land (LOTL), hands-on-keyboard style.

## Activity timeline

The sequence below separates the actor-history pivots from the confirmed Prinz Eugen ransomware activity, so readers can see what happened before and during the campaign.

| Date              | Event                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 9 Jul 2025        | Registers on XSS as avtokz; advertises a Vantage Finance database.                                                                               |
| 6 Sep 2025        | Joins Exploit as ROOTBOY.                                                                                                                        |
| 9 Sep 2025        | Sells data from a US/Canada driving-school software provider.                                                                                    |
| 8 Oct 2025        | Joins DarkForums.                                                                                                                                |
| 16 Nov 2025       | Advertises the 700Credit breach (~8.4M records) for ~\$2,500; states the data was sold after a failed extortion attempt under the name GERMANIA. |
| 27 Feb 2026       | Start of the Standard Bank + Liberty intrusion (~3-week dwell time).                                                                             |
| 16 Apr 2026       | Standard Bank breach posted on DarkForums; first public social-media report.                                                                     |
| From mid-Apr 2026 | Staged daily leaks of Standard Bank data escalate (5k to 25k to 50k to 100k rows/day) after the bank refuses to pay.                             |
| 3 May 2026        | Second Prinz Eugen victim: Transitions Pro Centre Val de Loire (France).                                                                         |
| 4 May 2026        | Formally claims the Standard Bank attack.                                                                                                        |
| 11 May 2026       | ThreatDown investigates an infected customer; encryptor analyzed.                                                                                |
| Late May 2026     | C2 panel stripped, typosquat DNS pulled and forum profile removed (infrastructure cleanup after the IP became public).                           |





ThreatDown EDR flagging RemotePC's PowerShell activity.

## Key takeaways

Prinz Eugen is an interesting addition to the current threat ecosystem because of its technical construction and the deliberate behavioral choices visible in the incident. The encryptor is new, written in Go, and built with enough care to prioritize high-pressure files, verify encrypted output, remove originals when instructed, and reduce forensic recovery opportunities before exiting.

The absence of a ransom note dropped on disk is a tactic we see more often among organized ransomware groups. By moving ransom communications entirely out-of-band (through direct email, phone contact, or dark-web victim portals), the actor reduces forensic artifacts and complicates automated detection of the extortion phase.

The use of legitimate RMM tooling (RemotePC) for post-exploitation persistence and lateral movement is consistent with a broader industry trend of ransomware operators abusing trusted software to blend into normal enterprise traffic and evade detection. Combined with the manual creation of a backdoor admin account, this indicates a hands-on-keyboard operator comfortable with LOTL techniques.

ThreatDown customers are protected through behavioral detections in both ThreatDown Endpoint Protection and ThreatDown EDR products, which surfaced the key attacker behaviors described in this report. Customers protected by ThreatDown would also have benefited from our anti-ransomware and Ransomware Rollback capabilities, allowing affected files to be restored automatically.

## Indicators of Compromise

---

The following Indicators of Compromise (IoCs) consolidate the network, leak-site, actor, and host artifacts discussed above.

### Network

---

The core network indicator is the host used for payload delivery, the observed panel, and the related domain infrastructure.

| Indicator               | Type                                  | Notes                                            |
|-------------------------|---------------------------------------|--------------------------------------------------|
| 212[.]80[.]7[.]74       | C2 / panel / payload host             | AS215439, Play2go International, Frankfurt, DE.  |
| stndrdbnk[.]cc          | Domain resolving to 212[.]80[.]7[.]74 | Standard Bank typosquat.                         |
| g-captchafestung[.]sbs  | Domain resolving to 212[.]80[.]7[.]74 | Fake-CAPTCHA / possible ClickFix lead.           |
| festung-e.duckdns[.]org | Domain resolving to 212[.]80[.]7[.]74 | Dynamic-DNS host observed between May 23 and 30. |

### Leak sites

---

The leak site indicators are included for reference and should be handled according to normal dark-web access controls.

| Address                                                          | Status                    |
|------------------------------------------------------------------|---------------------------|
| prinzfbjiazbrur4mje6mntjc4vydx3iatkkzycufoylqcoo4y7ppd[.]onion   | Active.                   |
| 6cudc5cqa2bjpwdhcwm2lj6dbqejjjqzeo6ipwvmbazr6cgu7vfk3dad[.]onion | Original, currently down. |

### Actor selectors

---

These selectors support attribution and pivoting. Host-based detection relies more directly on the file and endpoint artifacts below.

| Selector<br>type | Value                                                                            |
|------------------|----------------------------------------------------------------------------------|
| Handles          | ROOTBOY, avtokz, GERMANIA.                                                       |
| TOX              | 496187425B2944D73FBB17CAF3F9FD569B9ED3A08A497A8314CB4F27A51<br>E65081ACEE1E22F21 |
| Emails           | prinzeugen@mail2tor.co, standardbankcc@cock.li                                   |
| BTC              | bc1q2ztpcvqdaptej6uu2ywt9mrlatx6envu34rf0v                                       |

## Host / file artifacts

The host and file artifacts are the most directly useful indicators for endpoint hunting and incident-response review.

| Artifact type       | Value                                                                                                           |
|---------------------|-----------------------------------------------------------------------------------------------------------------|
| Payload             | servertool.exe                                                                                                  |
| Encrypted extension | .prinzeugen                                                                                                     |
| Go package          | scorched-earth-ausfc                                                                                            |
| File-header magic   | CHV1                                                                                                            |
| Crypto              | ChaCha20-Poly1305, 32-byte master key, KDF Argon2id to SHA256 to HKDF- SHA256, 1MB chunks, per-file random IVs. |
| Persistence         | net user admin germania /add                                                                                    |
| Self-delete         | cmd.exe /C ping 127.0.0.1 -n 2 > nul & del /F /Q ...\\Music\\servertool.exe                                     |
| RMM abuse           | RemotePC (IDrive) used for PowerShell staging.                                                                  |
| SHA-256             | 686213cc11d36af764de824801bced9366dfca3823fe0d51b752f74149bcf1f4                                                |
| PowerShell stager   | https://212[.]80[.]7[.]74/serverscan.ps1                                                                        |
| URLs                | https://212[.]80[.]7[.]74/stager/mini                                                                           |
|                     | https://212[.]80[.]7[.]74/stager/ps1                                                                            |

[ransomware](#) [rdp](#) [double extortion](#) [Go encryptor](#) [ChaCha20](#) [ROOTBOY](#) [anti-forensics](#)

## Cybercrime Has Gone Machine-Scale

AI is automating malware faster than security can adapt.

[Read the 2026 State of Malware](#)