

SEC Consult #ITSECX

Entropie im Sinkflug: rand() und srand()



an Eviden business

Entropie im Sinkflug: rand() und srand()

Präsentationsübersicht

Einführung

- Vorstellung
- Research Device
- Status
- Vorgehen

Schwachstellen, Exploits & Empfehlungen

- Improper Signature Verification of Firmware Upgrade Files
- Missing Protection Mechanism for Alternate Hardware Interface
- Predictable Session ID

CVEs / Responsible Disclosure Q&A und Diskussion

Entropie im Sinkflug: rand() und srand()

Einführung - Vorstellung

- Seit 2017 bei SEC Consult
- Bachelor
 - “Unternehmens- und IT-Sicherheit”
 - University of Applied Sciences Offenburg
- Master
 - IT-Security
 - FH Campus Wien

Entropie im Sinkflug: rand() und srand()

Einführung – Research Device

- Rittal IoT Interface
- “Vernetzung von Rittal Kühllösungen oder Sensoren zur Überwachung von physikalischen Umgebungsbedingungen“
- Research Projekt 2024 bei SEC Consult



Entropie im Sinkflug: rand() und srand()

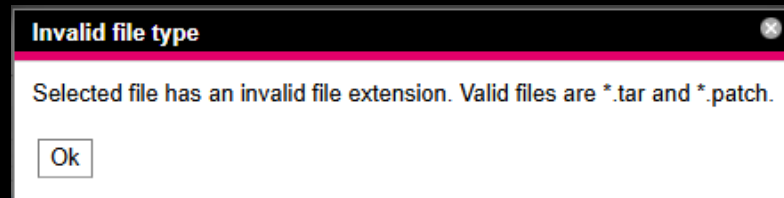
Einführung – Status

- Zugang zu Gerät
 - Physisch
 - Web Interface (Admin)
 - Kein Zugriff auf Dateisystem (IoT Interface)
- Zugriff auf Firmware-Datei des Herstellers (Download)
 - V6.19.00

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen

- Physische Überprüfung in unserem HW-Labor
 - Keine Auffälligkeiten
 - Debug Interface nicht offensichtlich
- Check des Web Interfaces
 - Check alter Schwachstellen
 - Session ID auffällig kurz
 - Firmware Upgrade verlangt nach .patch oder .tar Dateien



Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen

- Suche nach Dateien die `.patch` enthalten
- Code zur Verifizierung der Signatur
 - Ausschnitt aus `patch_verify.sh`

```
key="8ab4ce0f72bce9b3837281b792..."
filename=${filename_full%.*} # remove .patch

verify(){
    cd $patchhome
    for file in $(find . -type f \( ! -iname "rittal.sig" \) | sort ); do
        sha512sum $file >> rittal
    done
    echo $key >> rittal
    calcsig=$(sha512sum rittal)
    trustsig=$(cat rittal.sig)
    if [[ $calcsig == $trustsig ]]; then
        patchverify=OK;
    }
}
```

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh

- Speichere SHA512-Hash aller Dateien, außer `rittal.sig` in Datei `"rittal"`

```
key="8ab4ce0f72bce9b3837281b792..."

verify(){
    cd $patchhome
    for file in $(find . -type f \( ! -iname "rittal.sig" \) | sort ); do
        sha512sum $file >> rittal
    done
    echo $key >> rittal
    calcsig=$(sha512sum rittal)
    trustsig=$(cat rittal.sig)
    if [[ $calcsig == $trustsig ]]; then
        patchverify=OK;
    }
}
```


Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh – Key

- Füge statischen **key** an Ende der Datei "rittal" hinzu

```
key="8ab4ce0f72bce9b3837281b792..."
```

```
verify(){
    cd $patchhome
    for file in $(find . -type f \( ! -iname "rittal.sig" \) | sort ); do
        sha512sum $file >> rittal
    done
    echo $key >> rittal
    calcsig=$(sha512sum rittal)
    trustsig=$(cat rittal.sig)
    if [[ $calcsig == $trustsig ]]; then
        patchverify=OK;
    }
}
```

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh

- Erzeuge SHA512-Hash der Datei "rittal"
- Vergleiche, ob dieser Hash mit Inhalt der rittal.sig übereinstimmt
- Wenn Ja, patchverify=OK

```
key="8ab4ce0f72bce9b3837281b792..."

verify(){
    cd $patchhome
    for file in $(find . -type f \(! -iname "rittal.sig" \) | sort ); do
        sha512sum $file >> rittal
    done
    echo $key >> rittal
    calcsig=$(sha512sum rittal)
    trustsig=$(cat rittal.sig)
    if [[ $calcsig == $trustsig ]]; then
        patchverify=OK;
    }
}
```

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh

- Für Fake Signatur wird folgendes benötigt
 1. \$key
 2. rittal.sig
 3. Datei/Script, das nach erfolgreicher Verifikation ausgeführt wird

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh – run.sh

- Nach erfolgreicher Prüfung der Signatur wird run.sh ausgeführt

```
runpatch(){
    echo "run patch $filename_full($patchname)"
    cd $patchhome

    if [ -f run.sh ]; then
        source run.sh
    else
        echo "Error: No start script run.sh found"
    fi
    cd $home
    rm -r $patchhome
    echo "finished patch $filename_full($patchname)"
}
```

Entropie im Sinkflug: rand() und srand()

Einführung – Vorgehen – patch_verify.sh

- Für Fake Signatur wird folgendes benötigt
 1. \$key
 2. rittal.sig
 3. run.sh
- Möglich beliebige Dateien zu signieren
 - SHA512-Hashes der Dateien
 - "Secret" (key) ist bekannt
 - rittal.sig kann erstellt werden

Entropie im Sinkflug: rand() und srand()

Schwachstellen

Improper Signature Verification of Firmware Upgrade Files

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Improper Signature Verification of Firmware Upgrade Files – Exploit

1. SHA512sum aller Dateien (z.B. run.sh) für Fake Firmware in `rittal.fake`
2. Hardcoded Key an Ende von `rittal.fake` schreiben
3. SHA512sum von `rittal.fake` in `rittal.sig` schreiben
4. `rittal.fake` ggf. löschen
5. Tar der Fake Firmware Dateien und `rittal.sig` erstellen
6. In `<name>.patch` umbenennen
7. Firmware Upgrade durchführen (als Admin)
8. Gewinnen

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Improper Signature Verification of Firmware Upgrade Files – Demo

The image shows a desktop environment with two windows. The left window is a web browser displaying the Rittal IoT Interface. The browser's address bar shows the URL `192.168.1.144/index.html?~/.ITSECK`. The page content includes the Rittal logo, the slogan "Faster – better – worldwide.", and a login section. The login section has a header "Rittal IoT Interface" and a form with fields for "Username:" and "Password:". Below the form are two buttons: "Login" and "Login to Dashboard". To the right of the form, there is a list of fields: "Name : Name of the Unit", "Location : Location of the Unit", "Contact : Contact Person", and "IP Address: 192.168.1.144". At the bottom of the page, there is a navigation bar with five items: "ENCLOSURES", "POWER DISTRIBUTION", "CLIMATE CONTROL", "IT INFRASTRUCTURE", and "SOFTWARE & SERVICES". The footer of the page reads "2011-2023 © Rittal GmbH & Co. KG" and "FRIEDHELM LOH GROUP".

The right window is a terminal window titled "~/.ITSECK". The terminal shows the prompt `~/ITSECK >>` and is currently empty.

The Windows taskbar at the bottom shows the system tray with the time `00:30` and date `29.09.2024`. The system tray also includes icons for the Start menu, task view, and network. The bottom right corner of the image features the "Consult" logo and the text "an Eviden business".

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Improper Signature Verification of Firmware Upgrade Files - Lösungsempfehlung

- Asymmetrische Kryptographie
 - Firmware-Dateien werden mit privatem Schlüssel signiert
 - Signatur wird mitgeliefert
 - Private Key bleibt bei Hersteller
 - Public Key auf Gerät in Hardware Security Modul (HSM)
 - Wird verwendet, um Signatur zu prüfen
 - Mehrere Slots für Schlüssel
 - Falls der Private Key leaked wird
 - Schlüsselrotation
- Secure Boot
 - Sicherstellen, dass nur authentifizierte Firmware geladen wird

Entropie im Sinkflug: rand() und srand()

Schwachstellen

Missing Protection Mechanism for Alternate Hardware Interface

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Missing Protection Mechanism for Alternate Hardware Interface – Beschreibung

- Script `emmc_update_fromdir.sh` sucht u.A. nach `.patch` Dateien auf USB-Stick oder SD-Card
- Führt ohne Authentifizierung `patch_verify.sh` aus

```
[...]  
if [[ $UPDATE_FILE == *.patch ]];then  
[...]  
    bash /usr/sbin/patch_verify.sh "$WATCHDIR/${UPDATE_FILE}"  
[...]  
  
fi  
continue
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Missing Protection Mechanism for Alternate Hardware Interface – Exploit

1. .patch Datei aus vorigem Kapitel auf USB-Stick in root-Verzeichnis kopieren
2. Anstecken
3. Gewinnen

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID

Predictable Session ID

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Exkurs random

srand(unsigned int seed)

- Initialisiert rand()
- Typisch `srand(time(0))`

rand()

- `stdlib.h`
- Max Value: $2^{31}-1 = 2,147,483,647$
- Nimmt `int 1` als Seed, wenn nicht anders initialisiert
- NICHT kryptographisch sicher
- Deterministisch
 - Gibt (ohne `srand()`) immer 1804289383 zurück

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Authentifizierung

- Standard Zugangsdaten admin:admin
- Session nach 1800 Sekunden (30 Minuten) ungültig

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Authentifizierung Cont.

- HTTP Request bei Anmeldung gibt sehr kurze Session ID zurück
 - /cgi-bin/json.cgi ist für alles zuständig

| Request | Response |
|--|--|
| <pre>1 POST /cgi-bin/json.cgi HTTP/1.1 2 Host: 192.168.1.144 3 Content-Length: 63 4 Sec-Ch-Ua: "Not;A=Brand";v="24", "Chromium";v="128" 5 Sec-Ch-Ua-Platform: "Windows" 6 X-Requested-With: XMLHttpRequest 7 Accept-Language: en-US,en;q=0.9 8 Sec-Ch-Ua-Mobile: ?0 9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.6613.120 Safari/537.36 0 Content-Type: application/x-www-form-urlencoded 1 Accept: */* 2 Origin: https://192.168.1.144 3 Sec-Fetch-Site: same-origin 4 Sec-Fetch-Mode: cors 5 Sec-Fetch-Dest: empty 6 Referer: https://192.168.1.144/index.html? 7 Accept-Encoding: gzip, deflate, br 8 Priority: u=1, i 9 Connection: keep-alive 0 1 createSession= {"username":"admin","password":"admin","force":0}</pre> | <pre>1 HTTP/1.1 200 OK 2 Content-Type: text/plain;charset=iso8859-1 3 Strict-Transport-Security: max-age=15768000; 4 Content-Length: 246 5 Date: Thu, 01 Jan 1970 00:58:46 GMT 6 Server: lighttpd/1.4.55 7 8 {"result":0,"sessionId":1792516098,"user":{"name":"admin","group": "admins","groupId":1,"description":"Default Group","puAdministration":1,"filetransfer":2,"http":1,"console":1,"isLdap":0,"autologout":1800,"language":0,"unitType":0,"profileInet":0}} 9</pre> |

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Online Brute-Force #1

- Versuch, via HTTP Anfragen Session ID zu erraten
- Im Vergleich zu Offline Brute-Force extrem langsam
 - ca. 1000 Requests pro Minute
- Online Brute-Force gültiger Session (1800 Sekunden)
 - 2^{31} req / 1800 s = 1,2 Mio Requests pro Sekunde
 - Für Rittal IoT Interface unmöglich

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi

- Reverse Engineering der `json.cgi`
 - `json.cgi` wird für jeden Request neu aufgerufen, kein Fork
 - Ausschnitt des Codes zur Session ID Generierung

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – param_2

- Vergleich von originalen Session IDs und mit nachgebautem Code generierte Session IDs
 - Überproportionale Häufung von Kollisionen bei Wert 0 bzw. 1 für param_2
 - param_2 = Nutzer
 - Admin = 0
 - User = 1

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – pid

- getpid() gibt Prozess ID (PID) des aufrufenden Prozesses zurück
 - Settings auf IoT Interface 1-32768
 - PID 1 = Init oder system Prozess

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – seed

- srand wird mit rand() und PID initialisiert

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – Berechnung

- Session ID wird mit rand() und weiteren Werten berechnet

```
int FUN_00017384(undefined param_1, int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – Problem

- Wo ist das Problem?

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – Problem

- Wo ist das Problem?
 - `rand()` ist bei erstem Aufruf nicht initialisiert
 - Bei zweitem Aufruf Entropie nur durch PID

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```


Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – json.cgi – Problem

- rand() immer 1804289383
- param_2 = 0
- Entropie daher nur von PID (~32768 Werte)
- Bei fork() wäre rand() beim zweiten Aufruf bereits initialisiert

```
int FUN_00017384(undefined param_1,int param_2)
{
    int pid;
    int seed;

    pid = getpid();
    seed = rand();
    srand(seed + pid);
    return (rand() + 123 & 0xffffffff) + param_2 * 0x10;
}
```

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Exploit

- Erzeugen aller möglichen Session IDs
 - Mit Werten von 1 – 32768 für Variable `pid`
- Online Brute-Force #2
 - Server mit HTTP Anfragen fluten
 - Zuvor nicht sinnvoll, da >2Mrd Aufrufe in 30 Minuten unrealistisch sind
 - 32768 Anfragen sind durchaus möglich
 - Script sendet ca. 1000 Anfragen/Minute

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Demo

The screenshot displays a web browser window showing the Rittal IoT interface. The browser's address bar shows the URL `https://192.168.1.144/index.html?`. The page title is "Rittal – The System." and the Rittal logo is visible in the top right corner.

Overlaid on the browser is the Burp Suite Professional v2024.7.6 interface. The "Proxy" tab is active, and the "HTTP history" pane shows a list of intercepted requests. The selected request is a GET request to `https://192.168.1.144/index.html?`. The "Request" pane shows the raw HTTP request, and the "Response" pane shows the raw HTTP response. The response status is `HTTP/1.1 200 OK`, and the content type is `text/html`.

In the foreground, a Notepad window titled "Untitled - Notepad" is open, showing a blank document. The status bar at the bottom of the Notepad window indicates "Ln 1, Col 1", "300%", "Windows (CRLF)", and "UTF-8".

On the right side of the browser window, a timer is visible, showing `00:00:00,00` with "Std", "Min.", and "Sek." labels. Below the timer, there are three circular buttons: a play button, a stop button, and a refresh button.

At the bottom of the browser window, a navigation bar contains several menu items: "ENCLOSURES", "POWER DISTRIBUTION", "CLIMATE CONTROL", "IT INFRASTRUCTURE", and "SOFTWARE & SERVICES". Below this bar, the text "FRIEDHELM LOH GROUP" is visible.

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Predictable Session ID – Lösungsempfehlung

- Verwendung kryptographisch sicherer Zufallszahlengeneratoren (CSPRNG)
 - z.B. getrandom() erhält Entropie von urandom
- Keine Vorhersagbarkeit der Session IDs
- **Ausreichende Länge der Session IDs**

- Best Case (falls möglich):
 - Verwendung von Frameworks für die Authentifizierung
 - OpenID Connect / OAuth 2.0
 - Spring Security (Java)
 - Django REST Framework (Python)
 - ...

Entropie im Sinkflug: rand() und srand()

Schwachstellen – Wie könnte ein Angriff aussehen?

- Angreifer erlangt Zugriff auf internes Netzwerk
- Verschlüsselt Dateien / Zieht wertvolle Daten ab
- Für Persistenz
 - Brute-Force der Session ID
 - Einbauen einer Backdoor über Firmware Upgrade in Rittal IoT Interface
- Opfer setzt Active Directory neu auf
- Angreifer erhält erneut Zugriff über Backdoor des IoT Interfaces

Entropie im Sinkflug: rand() und srand()

CVEs / Responsible Disclosure

- 1) Improper signature verification of firmware upgrade files (CVE-2024-47943)
 - 2) Missing Protection Mechanism for Alternate Hardware Interface (CVE-2024-47944)
 - 3) Predictable Session ID (CVE-2024-47945)
- Kontaktaufnahme mit Hersteller gemäß Responsible Disclosure im Juni 2024
 - Patch V6.21.00.2 verfügbar seit 2024-09-30
 - Advisory-Release in KW42 geplant

Entropie im Sinkflug: rand() und srand()

Q&A und Diskussion

Fragen?