1-1-2023

CamDec: Advancing axis P1435-LE video camera security using honeypot-based deception

Leslie F. Sikos
Edith Cowan University

Craig Valli
Edith Cowan University

Alexander E. Grojek
Edith Cowan University

David J. Holmes
Edith Cowan University

Samuel G. Wakeling
Edith Cowan University

See next page for additional authors

10.1007/s11416-023-00463-4
Sikos, L. F., Valli, C., Grojek, A. E., Holmes, D. J., Wakeling, S. G., Cabral, W. Z., & Karie, N. M. (2023). CamDec: Advancing axis P1435-LE video camera security using honeypot-based deception. *Journal of Computer Virology and Hacking Techniques, 19*(4), 565-577. https://doi.org/10.1007/s11416-023-00463-4
This Journal Article is posted at Research Online.
https://ro.ecu.edu.au/ecuworks2022-2026/2005

Authors
Leslie F. Sikos, Craig Valli, Alexander E. Grojek, David J. Holmes, Samuel G. Wakeling, Warren Z. Cabral, and Nickson M. Karie

Journal of Computer Virology and Hacking Techniques
https://doi.org/10.1007/s11416-023-00463-4

ORIGINAL PAPER

CamDec: Advancing Axis P1435-LE video camera security using honeypot-based deception

Leslie F. Sikos[1] · Craig Valli[1] · Alexander E. Grojek[1] · David J. Holmes[1] · Samuel G. Wakeling[1] · Warren Z. Cabral[1] · Nickson M. Karie[2]

Received: 12 May 2022 / Accepted: 9 January 2023

Abstract
The explosion of online video streaming in recent years resulted in advanced services both in terms of efficiency and convenience. However, Internet-connected video cameras are prone to exploitation, leading to information security issues and data privacy concerns. The proliferation of video-capable Internet of Things devices and cloud-managed surveillance systems further extend these security issues and concerns. In this paper, a novel approach is proposed for video camera deception via honeypots, offering increased security measures compared to what is available on conventional Internet-enabled video cameras.

Keywords Video stream security · Video tampering prevention · Honeypot · Cyberattack · Intrusion detection · Video data privacy · Network forensics

1 Introduction

Recent advancements in cloud-managed video cameras, video compression and streaming, and artificial intelligence applications in camera surveillance have enabled automated anomaly detection [23], pose estimation [18], real-time movement tracking [24], person (re)identification [31], and many more automated functions.

However, Internet-connected video surveillance systems have well-known security and privacy concerns, which are particularly important, given the large number of cloud-managed video surveillance systems [1]. The range of IP camera attacks includes software attacks (weak access control and authentication, insufficient transport layer protection, denial of service, cross-site scripting (XSS), cross-site request forgery (CSRF), path traversal, information leakage via file disclosure, command injection, buffer overflow, reverse engineering, unsigned/unverified upgrade); software/hardware attacks (data exfiltration); hardware attacks (debug protocol attacks, bootloader attacks, time-of-check to time-of-use (TOCTTOU) attacks, unsigned/unverified upgrade); optical attacks (command and control, data exfiltration), camera blinding (dazzling)); and—in case of wireless cameras—RF/wireless attacks (RF jamming, denial of service, eavesdropping) [9]. The login page of the web interface of IP surveillance and security cameras and other files, such as CGI scripts available through URLs for status requests, may leak camera model version, firmware, timestamps, serial number, P2P port number, Wi-Fi SSID, etc. even without authentication [19], while malware might exploit camera vulnerabilities to access sensitive information, such as passwords and PIN codes [15].

Given the array of potential cyberattacks against IP cameras (that are enabled by a variety of vulnerabilities and exploits, as seen, for example, in the Directory of Video Surveillance Cybersecurity Vulnerabilities and Exploits of IPVM),[1] an increasing number of camera vendors implement various security countermeasures—see the use of SSL certificates and the management data encryption on Cisco Meraki cameras,[2] for example.

Many commercial network cameras systems come with dedicated software applications for monitoring and control [12]. These are important from the security perspective because of the following reasons:

✉ Leslie F. Sikos
l.sikos@ecu.edu.au

1 Edith Cowan University, Perth, Australia

2 Cyber Security Cooperative Research Centre, Perth, Australia

1 https://ipvm.com/reports/security-exploits

2 https://meraki.cisco.com/products/smart-cameras/

Published online: 10 February 2023

- They can have security vulnerabilities specific to the associated product or product line, which might rely on vendor-specific patches not released for years.
- The features of these software can potentially characterize a device, indicating a product line or even a particular camera model. This is important in determining how to emulate a camera with a honeypot to be deceptive, because leaving out any fundamental property and including features that are not supported by a camera would reveal to adversaries that the device is not what it looks like. Return values of executed commands and the faithful reproduction of actions such as camera zoom are actively used by attackers for fingerprinting IP cameras [27].
- The network traffic generated and utilized by these tools, along with the (often uncommon) TCP/UDP port number(s) used by them, are crucial when setting up security measures in the infrastructure.

2 Related works

Vulnerabilities of IP cameras include, but are not limited to, default credentials that can be used for brute force attacks, exposed TCP timestamps that allow attackers to compute uptime (leading to further attacks), and allowing OS fingerprinting [28]. A variety of freely available software tools can be used for the cyber-reconnaissance of IP cameras, potentially obtaining device data, such as IP address, hostname, MAC address, vendor, and server information, and target camera systems via the Virtual Switch System (VSS) port used for video streaming [10]. Sabotage intervention has to be timely, not only the logical tampering of recorded video footage, but also the physical tampering of camera devices, whether via occlusion, defocus, or displacement, should be automatically detected [25].

Baseline security countermeasures for IP cameras include, but are not limited to, changing default usernames and passwords, preventing information forwarding to third-parties, avoid port-forwarding to the camera, monitoring network traffic for spikes, and keeping camera firmware up to date. However, these do not protect from sophisticated cyberattacks, which require additional countermeasures. For example, Vempati et al. [30] proposed an approach that employs a micro-firewall rule to detect known traffic and allows video streaming even during cyberattacks, such as distributed denial-of-service (DDoS) attacks.

CameraObscura[3] aims to mimic common features of IP cameras, such as camera stream, login, and firmware upload, thereby protocolizing botnet actions. It provides JSON-configurable routes to simulate logins and new firmware uploads according to vendor specifications, configurable headers to simulate a vulnerable web server, a web interface, and logging (text and JSON), as well as payload dumps (e. g., on fake firmware upload or `POST` with file).

The *SIPHON* architecture is an implementation of a distributed IP camera honeypot, featuring multiple IP cameras, a network video recorder (NVR), and an IP printer [14]. In SIPHON, camera locations are spoofed using wormholes based on instances in cloud services. These instances are deployed in different cities around the world, which can mislead adversaries using software tools such as *Shodan*[4] or *Fofa*[5] to identify IoT device locations, but the traffic is actually directed from the wormholes to a single location. This is particularly important from the deception point of view, because having any indication of a honeypot could discourage adversaries from performing—manual—attacks, considering all the factors that might impact the probability of an attacker giving up attacks, which can be estimated by a monotonic function of the form

$$p_g = \frac{h\left(c\left(1 + r_h p_h\right)\right)}{\left[\left(c_r + c_c\right)\left(1 - p_h\left(1 - v_h\right)\right)\right]} \qquad [22]$$

where h is a sigmoidal function (e.g., hyperbolic tangent), c_r is the average (time) cost of reconnaissance, c_c is the total cost of subsequent random system compromise, c is the perceived remaining cost to the attacker to achieve their goals, p_h is the probability that the system is a honeypot as perceived by the attacker, r_h is the rate of damage to the attacker (where damage is assumed to be proportional at this rate to the time the attacker spends in the honeypot), and v_h is the ratio of the attacker's perceived value of compromise (linear weighting if the attacker is unsure whether the system is a honeypot).

While SIPHON uses physical IoT hardware for the honeypots, along with the VPN connections and virtual machines to handle the networking, the developers also exposed a low-interaction honeypot running Trendnet camera emulator and observed the results compared to their high-interaction honeypots running physical IoT hardware. Note that on average, SSH sessions with the low-interaction honeypots lasted 30 seconds compared to one minute with the high-interaction honeypots.

As only one low-interaction honeypot is exposed with SIPHON, compared to the number of high-interaction physical IoT honeypots along with the different the geographical locations, this data is inconclusive. The authors found that the geographical location of the IP address of the exposed system did make a difference, therefore it's possible that the low-interaction honeypot may have generated different data to the equivalent or similar high-interaction honeypots due to

[3] https://github.com/CMSecurity/CameraObscura

[4] https://www.shodan.io

[5] https://fofa.so

changes in the geographical location of the IP address they were exposed at.

Furthermore, as the devices were not modified, attackers could use the wireless capabilities of the devices to attack other nearby networks, or discover their real geographical location compared to the location shown by their IP address. Another approach, presented in [27], is to remove the Wi-Fi oscillators from their devices so that attackers cannot leverage the Wi-Fi capabilities of the honeypots.

While the authors in [14] discuss possible methods of device compromise detection and restoration for physical IoT devices, they do not discuss which methods they actually used in their research, and how regularly they restored and reset the firmware on their devices. This means that a previously infected device may still be in use when new attackers connect to it, meaning they are not connecting to a fresh honeypot each time and may not act in the same way upon knowing the device is already infected, or the device may not respond the same way.

Various parameters of cyberattacks can be utilized in machine learning to iteratively refine honeypot deception. For example, by using reinforcement learning, the interaction with an attack sequence can be prolonged via a transition reward function of the form

$$r_t(s_i, a) = \begin{cases} 1 & \text{if } i \in Y \\ 0 & \text{otherwise} \end{cases} \qquad [11]$$

where s_i is the state and a is the action, rewards the learning agent if a bot command is input string i, comprised of bash commands (L, known Linux commands, such as `wget`, `cd`, `mount`, and `chmod`), customized commands (C, commands executing downloaded files), or compound commands (CC, multiple commands with bash separators/operators), which form state set $Y = L \cup C \cup CC$.

3 Known P1345-LE vulnerabilities

Video camera vulnerability exploitation, similar to other hardware devices, strongly depends on the version of the firmware installed. A report released in 2018 by the security team at Vdoo has identified seven zero-day vulnerabilities that, at the time, affected 390 models of AXIS Communications' IP cameras [21]. The P1435-LE on firmware older than v6.50.2.3 was noted as one of the models affected by the vulnerabilities identified [3]. This means that if an attacker manages to obtain a camera's IP address on the Web and theoretically string multiple vulnerabilities together, a camera could be fully compromised. From our viewpoint, knowing the details of the vulnerabilities disclosed enables us in further accurately spoofing the IP camera within the honeypot environment. Since our honeypots do not necessarily have to be on the latest firmware version, they can represent IP cameras that are rarely maintained, often though a set-and-forget approach. These vulnerabilities would allow attackers to bypass the IP camera's login page and take full control of the camera's configuration and in theory the honeypots, including the following:

- Accessing the live view and freezing the video stream;
- Controlling the camera's point of view;
- Disabling the motion detection capabilities;
- Disabling the camera entirely.

An attacker can also take more malicious actions through:

- Modifying the camera's firmware;
- Adding the camera to a botnet;
- Using the camera as a network infiltration point;
- Using the camera to perform other malicious actions (DDoS attacks, crypto mining).

All Axis IP camera firmware are built on top of an underlying Linux operating system, each modified for its particular camera's specifications. As Linux is utilized, the camera makes use of common services including Apache, GStreamer, and modified versions of FTP, SSH, and Telnet to provide core functionality. However, this poses a problem because new vulnerabilities can be present in the firmware core service dependencies. From Vdoo's analysis of the firmware files and interaction with the camera's front-facing interface, multiple issues where identified, ranging from authorization bypass and shell command injection to crashing the web server [21]. These vulnerabilities are identified through the CVE IDs:

- *CVE-2018-10658*:[6] A memory corruption vulnerability discovered in multiple models of Axis IP cameras, which causes a denial of service (crash). The crash arises from code inside the `libdbus-send.so` shared object (or similar).
  This vulnerability allows an unauthenticated adversary to send a `dbus-request` (via the `/bin/ssid .srv` interface) with a specially crafted string to crash the SSID service. This crash arises from the code inside `libdbus-send.so` shared object (or similar).
  Because the crashes also occur by directly invoking `"/usr/bin/dbus-send"` with a similar string, this may affect other processes that include this code.
- *CVE-2018-10659*:[7] A memory corruption vulnerability discovered in multiple models of Axis IP cameras, which

[6] https://nvd.nist.gov/vuln/detail/CVE-2018-10658
[7] https://nvd.nist.gov/vuln/detail/CVE-2018-10659

allows remote attackers to cause a denial of service (crash) by sending a crafted command, which will result in a code path that calls the `UND` undefined ARM instruction.
This vulnerability allows an unauthenticated adversary to send a specially crafted command (via the `/bin/ssid .srv` interface) that will result in a code path that calls the `UND` undefined ARM instruction (and possibly a similar scenario in MIPS or other architectures' cameras) that causes the process to crash.

– *CVE-2018-10660*:[8] A shell command injection vulnerability was discovered in multiple models of Axis IP cameras.
To take advantage of this vulnerability, one must have permissions to change certain `parhand` parameters. This can be achieved by either 1) achieving/having administrator privileges (by using the CGI interface), 2) executing code inside the `upnp` daemon, or 3) finding other ways to control certain `parhand` parameters (see CVE-2018-10662).
The `parhand` parameter handler is responsible for fetching, storing, and changing many of the device's internal parameters. When a user sets a parameter through the web interface, the relevant CGI script (`param.cgi`) forwards the set-parameter request to the `parhand` binary, which checks access rights, and stores the parameter's value in the relevant configuration file.
Some of the parameters are used for feeding shell scripts, and are defined as shell-mounted (`mount = "Shell{`�`}"` in the `parhand` configuration file). The parameters' values are parsed by the `parhand` Shell-Parser, which does not sanitize special shell characters, and also does not quote the parameters' values.
The shell scripts directly execute the configuration file (for the purpose of including the configuration parameters). By setting the parameter's value with a semicolon (`;`), we were able to inject arbitrary shell commands with root privileges.

– *CVE-2018-10661*:[9] A bypass of access control vulnerability discovered in multiple models of Axis IP cameras.
This vulnerability allows an adversary to bypass the web server's authorization mechanism by sending unauthenticated requests that reach the `/bin/ssid`'s `.srv` functionality. This vulnerability resides in `modauthzaxisgroupfile.so`, a custom authorization module for Apache `httpd` that was written by the vendor.
Requests to a world-readable file that are followed by a backslash and end with the `.srv` extension (e.g., `http://CAMERA_IP/index.html/a.srv`) are treated by the authorization code as standard requests to the `index.html` and thus granted access, while the requests are also treated as legitimate requests to an `.srv` path, and are thus handled by the `.srv` handler simultaneously.

– *CVE-2018-10662*:[10] An exposed insecure interface vulnerability discovered in multiple models of Axis IP cameras.
Legitimate requests that reach the `/bin/ssid`'s `.srv` functionality can choose one of several actions by setting the action parameter in the request's query string. `dbus` allows the user to invoke any `dbus` request as root (the `uid` and `gid` of the `/bin/ssid` process), without any restriction on the destination or content. Due to the `dbus` request originating from a root process, unrestricted access is granted to many `dbus-services`' interfaces.
While the `dbus` interface in `/bin/ssid` only serves the purpose of fetching specific values from some specific `dbus`-enabled services, it exposes a much broader functionality. Thus, this interface gives users the ability to control any of the device `parhand` parameters' values. Control can be achieved by sending `dbus`-requests to invoke `policykit_parhand` process' `dbus`-interface (PolicyKitParhand) functions.

– *CVE-2018-10663*:[11] An incorrect size calculation vulnerability discovered in multiple models of Axis IP cameras.
The `return_page` and `servermanagerreturn page` query string parameters in `/bin/ssid`'s `.srv` functionality are controlled by the user, and returned back to her in the response to the user's request. When dealt with in the response-building code, these fields' values are trimmed to a size of 0x200 bytes and copied to a malloced 0x200-bytes space by using the safe `snprintf_chk` function. Then the return value of the `snprintf_chk` function (supposedly their length) is saved in a `struct` member variable for later calculating the response's content length.
The vulnerability lies when the return value of the `snprintf_chk` function is the "the number of characters that would have been written if *n* had been sufficiently large." This makes the calculated content-length larger than the actual data buffer, and as a result, extra bytes from memory are leaked in the response.

– *CVE-2018-10664*:[12] A memory corruption vulnerability discovered in the `httpd` process in multiple models of Axis IP cameras.

[8] https://nvd.nist.gov/vuln/detail/CVE-2018-10660
[9] https://nvd.nist.gov/vuln/detail/CVE-2018-10661
[10] https://nvd.nist.gov/vuln/detail/CVE-2018-10662
[11] https://nvd.nist.gov/vuln/detail/CVE-2018-10663
[12] https://nvd.nist.gov/vuln/detail/CVE-2018-10664

This vulnerability allows an unauthenticated adversary to crash the `httpd` process, causing (at least) a black screen for viewers that were already logged to the camera using the web interface with default settings.

Upon disclosure from Vdoo, AXIS promptly published their own security advisory ACV-128401[13] combining all seven vulnerabilities and released firmware updates for all effected models. From Vdoo's research, they do not believe that any of the vulnerabilities have been exploited in the wild, leading to no known deliberate compromise, malware, security or data privacy threat. However, they do suggest the following recommendations to be implemented in newer firmware as a result of their findings:

- *Improved privilege separation:* It was identified that inadequate separation of code execution allowed functions to be executed with root privileges from an already running root process. Ensuring that an executed system processes is limited to only essential outside calls and variables is critical. This involves segmenting blocks of code execution and restricting necessary privileges (root) as required for that segment. Less reliance on root execution makes it difficult for an attacker to escalate their privilege and move laterally within the system.
- *Adequate input sanitization:* It was determined that no input validation was performed on all user input from the front-facing web interface when configuring the camera's settings. This leaves potential invalid characters from not being removed prior to being saved to the camera's configuration file. Invalid characters can cause problems during script execution and potentially lead to bugs and exploits. Input sanitization is critical to ensure that only valid characters are accepted as user input.
- *Reduced reliance on shell scripts:* The firmware makes use of multiple shell scripts that process user input from the front-facing web interface. These inputs are then saved to that camera's particular configuration file. If improper input is saved and then executed by the shell script, arbitrary commands can be executed as root. By reducing or better eliminating the use of shell script executing as root, it becomes less likely that arbitrary malicious code can be executed.
- *Encryption of firmware binary:* As the camera firmware is unencrypted, anyone with technical know-how can view, extract, and analyze the firmware for anything of interest. This includes file structures, code/scripts and variable/-function names to identify how the firmware works and weak areas that can be exploited. Encryption can be implemented in two method: 1) encrypting the firmware binary itself, or 2) security through obscurity.

To mitigate all known vulnerabilities and potential threats, all affected Axis cameras have to be maintained with the latest firmware released.

4 A novel approach for advanced video camera security via honeypots

The purpose of this research paper is to establish an approach, called *Camera Security Using Deception (CamDec)*, whereby a Cowrie-based honeypot spoofs a hardware-based instance of a physical device. Such devices may include enterprise-grade network hardware (routers, switches, access points), IP cameras, IP phones, video and teleconferencing equipment, and Internet-of-Things (IoT) system-on-a-chip (SoC)/ embedded devices, just to name a few. Traditionally, honeypots have been built to simulate unconfigured, poorly maintained devices to entice attackers in enterprise networks. However, this research aims to take this a step further by seeking to identify methods of spoofing IoT hardware, firmware, and software within a virtual Cowrie honeypot environment.

Specifically, this paper focuses on spoofing an Axis P1435-LE IP-based camera. Axis Communications is a leading global manufacturer and vendor of network video surveillance and analytics, access control, audio systems equipment, and communication platforms. The reason for selecting this specific IP camera brand is its global deployment in a wide variety of enterprises and commercial environments, including both the public and private sectors.

The aim of our research is to develop, design, and build an implementation of a Cowrie-based honeypot that efficiently spoofs an Axis IP camera instance. This research focuses specifically on the Axis P1435-LE IP camera, and hardware, firmware, and software analyses, enumeration, and fingerprinting conducted on it. Based on these findings, the reverse engineering of the camera's default front-facing public interface is attempted to identify the device's unique traits for accurately mimicking them.

At the time of writing, the latest firmware release for P1435-LE is version 9.80.2.2 LTS (long-term support).[14] The P1435-LE is described by the manufacturer as a compact bullet camera suitable for any light condition, with built-in infrared LEDs, 1080p recording at 50/60 fps, and support for two lenses (wide and telephoto) [4].

4.1 Camera configuration

As an initial step, a factory reset was performed on the P1435-LE, and the default configuration was left unchanged. Once connected to the Local Area Network (LAN), the camera

[13] https://www.axis.com/files/faq/Advisory_ACV-128401.pdf

[14] https://www.axis.com/ftp/pub_soft/MPQT/P1435-LE/latest/relnote.txt

was accessed via the device's web interface using its default credentials. This access also enabled the configuration of the device, with access to functions such as IP addressing, DNS, date/time, recording settings, and port enabling/disabling.

4.2 Automated Cowrie installation

In the proposed solution, a combination of tools is used to automatically create and deploy Ubuntu virtual machines that host Cowrie honeypots. *Packer*[15] is used to download and install Ubuntu 18.04 LTS, together with *Ansible*[16] to automatically update and install dependencies for virtual machines, before using a shell script to configure the Cowrie honeypot instances.

Building on related works of other researchers, Packer, Ansible, and associated shell scripts have been selected to be used for this project. The reasoning behind this design is supported by efficient building and rapid deployment of an unmanaged image across multiple VM hosts.

In addition, this solution ensures the virtual honeypot instances' integrity, and that they meet the specific operational requirements of the project build and application. The use of pre-built Vagrant machines[17] or Docker containers[18] was discarded due to project scope and because of their vulnerability to malicious creation and modification.

The security concerns inherent in Docker provide an edge to attackers [29] through denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. Furthermore, the possibility of privilege escalation is heightened when employing Docker containers, as they require operational access to the Linux kernel [16].

Our project approach supports custom configurations, such as specific camera model templates, so that honeypots can run bespoke virtual environments rather than using default configurations that might otherwise be more readily detected [6,7]. This approach reduces reliance upon third-party image updates, such that the desired version of software can be installed to meet specific application requirements, together with that of the operating system.

Furthermore, by leveraging the technology of virtualisation, in the event of a honeypot compromise, or each time the honeypot is accessed, the administrator can quickly implement a "destroy and re-deploy" action. In contrast, physical devices would need to be manually re-flashed with the correct firmware, assuming that the compromise did not damage the underlying physical hardware.

[15] https://www.packer.io

[16] https://www.ansible.com

[17] https://www.vagrantup.com

[18] https://www.docker.com

4.3 Ensuring plausibility

Plausibility is a critical aspect of any honeypot configuration, where the fundamental goal of the honeypot is to mimic a vulnerable camera, and to attract and extend attacker interaction for as long as possible. This mimicry ensures that sufficient evidence can be collected about attackers and their malicious actions. The ability of the honeypot to mimic a hardware-based instance of any physical device is of paramount importance.

There remains the potential for further research in this field with a particular focus on improving honeypot believability throughout the attacker honeypot interaction.

The P1435-LE supports advanced features, including 1080p video recording, Lightfinder,[19] Wide Dynamic Range (WDR), OptimizedIR (Axis proprietary technologies) [2], and includes support for SSH and Telnet out of the box. To define a clear scope for our research, the constraints under which the P1435-LE camera operates had to be outlined. For this reason, the default configuration and the most common services of the camera have been analyzed.

In order to determine the most common ports open on the Axis P1435, the project team turned to IoT search engines Censys and Shodan. Used by attackers and penetration testers alike [8], Shodan has become a favorite cyber-toolkit, because it can identify and index public-facing IP devices. Access to the IoT search engines is through a web browser, permitting the search engines to discover device types and models of interest that have a public-facing presence online. The search engines work by scanning the Internet using a custom scan algorithm, leveraging user-entered search criteria, and parsing the banners returned by the device as a result of the search engine scan. Search terms can be concatenated and filtered, thereby permitting a customizable search functionality.

For the purposes of this project, the search was defined using the make and model of the camera of interest. The results are then displayed through the web browser interface delivering information such as the following:

(a) IP address
(b) Hostname
(c) ISP
(d) Open ports
(e) Protocols
(f) Device host country
(g) Device host city
(h) Detected banner(s)
(i) Date of last update

[19] Axis Lightfinder is a technology to deliver high-resolution, full-color video with a minimum of motion blur even in near darkness.

In addition, Shodan can provide an assessment on the likelihood that the device is a honeypot. By comparing previously defined characteristics of known honeypots, Shodan can estimate the probability that a scanned IP address is a honeypot and describe that probability as a so-called "honeyscore." A honeyscore assigns a numeric value between 0.0 and 1.0 to the IP address.

While the P1435-LE does have added support for SSH (port 22) and Telnet (port 23), these services are not enabled by default. Only FTP (port 21) is enabled. While these additional services can be opened by the administrator of the camera, in a commercial setting this is unlikely to be the case, due to the security risk associated with leaving ports such as SSH and Telnet open. Performing a Nmap scan on the P1435-LE also confirms the default camera configuration (see Listing 1).

```
PORT    STATE SERVICE VERSION
21/tcp open ftp Axis P1435-LE
   Network Camera ftpd 9.80.1
   (2020)
80/tcp open http Apache httpd
   2.4.41 ((Unix) OpenSSL/1.1.1d)
| http-robots.txt: 1 disallowed
   entry
|_/
|_http-server-header: Apache
   /2.4.41 (Unix) OpenSSL/1.1.1d
|_http-title: AXIS
554/tcp open rtsp GStreamer rtspd
|_rtsp-methods: ERROR: Script
   execution failed (use -d to
   debug)
MAC Address: 00:40:8E:DE:34:31 (
   Axis Communications AB)

Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel
   :3 cpe:/o:linux:linux_kernel:4
OS Details: Linux 3.2-4.9
Network Distance: 1 hop
Service Info: OS: Linux; Device:
   webcam; CPE: cpe:/h:axis:p1435
   -le_network_camera,
cpe:/o:linux:linux_kernel
   :4.9.206-axis5
```

Listing 1 Nmap scan results of a P1435-LE default configuration

As determined by the Nmap scan, in conjunction with the results from Censys and Shodan, the scope for masquerading the Cowrie honeypot configuration will be limited to the following open ports of the P1435-LE camera:

- Port 21 (FTP): Axis P1435-LE Network Camera ftpd 9.80.22 (2020)
- Port 80 (HTTP): Apache httpd 2.4.41 ((Unix) OpenSSL/1.1.1d)
- Port 554 (RTSP): GStreamer rtspd

4.3.1 Mimicking the camera's file system

The camera's OS architecture is built upon the Yocto project's Poky OpenEmbedded framework, which was described by [5] as a collaboration project to develop templates for various hardware platforms. Its functionality enables developers to efficiently prototype software projects.

The Axis P1435 camera file system is based on a customized version of BusyBox v1.31.0, modified to perform camera-specific functions from an embedded Linux environment. BusyBox is described as a multi-call binary that combines minimalist versions of common Unix utilities in a small executable. Licensed under GPLv2, BusyBox is open source, has an active community participation, and provides access to the latest versions available through the Busybox website.[20] In the case of P1435, BusyBox is shipped with over 100 functions, including `dd`, `echo`, `find`, `fdisk`, `grep`, `head`, `tail`, `uname`, `inetutils`, `modutils`, `net-tools`, `sysvinit`, and `tar`, among others.

Mimicking of the camera file system was achieved using the built-in Cowrie `createfs` utility. The source files used were derived from either a Busybox binary or alternatively from the camera firmware binary file, where the use of one or the other was dictated by the specific honeypot deployment deliverable. For example, by downloading an earlier firmware version for a camera from the OEM website, it is possible by extracting the squashfs from the firmware .bin file, to use this to create a "vulnerable-looking" file system within the honeypot. Known Axis camera CVE vulnerability lists are available on the company website, however, most firmware updates cannot be automated, and therefore the responsibility for updating the firmware remains with the end user, which may result in overlooked security updates. This can be witnessed by IoT search engines, which often reveal outdated firmware.

4.3.2 Utilizing the user accounts

The Axis P1435-LE camera is delivered with three disabled factory default accounts, those being:

- Administrator: can change settings, access recordings, and live view.
- Operator: can access recordings and live view.
- Viewer: can access live view.

[20] http://busybox.net/downloads/

These accounts can then be enabled as part of the camera configuration process and assigned appropriate passwords at the discretion of the camera deployment engineer. Once configured and enabled, the camera `/etc/passwd` and `/etc/shadow` file contents will need to be copied and imported in their respective locations within the honeypot, thereby accurately mimicking the user account structure of the donor camera.

Since September 2018, the credentials attackers can use to connect to Cowrie are stored in /etc/userdb.txt. Previously, this file was located in /data/userdb.txt and came as a part of the default Cowrie build. However, since September 2018, the `userdb.txt` must be created and configured from `/etc/userdb.example` as provided in the Cowrie build.

The credentials for the previously created camera accounts (Administrator, Operator, and Viewer) can be copied and added to the honeypot's `userdb.txt` file. In addition, by editing the `userdb.txt`, it is possible to specify user account/password combinations that can be used by attackers to gain access to Cowrie, and exclude all others. For this purpose, we used the top 20 Sucuri Wordpress honeypot user account/password combination captures, and applied them to each of the Administrator, Operator, and Viewer user accounts. This can extend an attacker's login interaction attempts with the honeypot, thereby providing the opportunity for generating a more extensive login record.

4.3.3 Replicating camera functions

Some of the ports used by P1435-LE can be reproduced on Cowrie, as shown in Table 1.

The possibility to replicate the P1435-LE functions from the honeypot is determined by the feasibility of spoofing the live feed from the camera in this environment. According to the Nmap scan performed, the RTSP stream service was identified as GStreamer.[21] However, for our purposes, VLC,[22] a free and open source multimedia player, was utilized as the RTSP server on the honeypot for two reasons.

Firstly, it had to be determined if service spoofing is indeed possible and the mechanisms of how Nmap detects services in order to accurately match it to the service aiming to be spoofed. Secondly, to aid with easier mass-scale honeypot deployment using VMs with sets of standardized software preloaded that can be modified by the user to appear as something else to Nmap.

As part of this research, we attempted to spoof a live video feed through VLC. Feeding VLC a video file of earlier recorded footage has the following benefits:

1. it looks believable;
2. the first and last frames loop together at a clean cut; and
3. the file used has the same video properties of which the P1435-LE would record at.

For our purposes, we used a ten-minute recording from the P1435-LE with the timestamp removed (which would compromise the authenticity of the stream when the video stream is played in a loop). This was done in an Ubuntu 18.04 testing environment as follows:

1. Install VLC on Ubuntu:
```
sudo apt install vlc
```
2. Set VLC to use root (geteuid to getppid):
```
sudo sed -i 's/geteuid/getppid
    /' /usr/bin/vlc
```
3. Create spoofstream bash script:
```
sudo nano /home/<user>/<
    directory>/<to>/spoofstream.
    sh
```
4. Copy the command into the spoof stream bash script:
```
#!/bin/bash
cvlc /home/<user>/<directory>/<
    to>/<video_file.extension> :
    sout=#rtp{sdp=rtsp://:554/
    axis-media/media.amp} --
    repeat --sout-keep
```
5. Change permissions of the `spoofstream` bash script:
```
sudo chmod 774 /home/<user>/<
    directory>/<to>/spoofstream.
    sh
```
6. Create a service file:
```
sudo nano /etc/systemd/system/
    axisspoof.service
```
7. Copy the configuration into the service file:
```
[Unit]
Description=Axis Spoof Livefeed

[Service]
WorkingDirectory=/home/<user>/<
    directory>/<to>/<
    working_directory>
ExecStart=/home/<user>/<
    directory>/<to>/spoofstream.
    sh

[Install]
WantedBy=multi-user.target
```

21 https://github.com/GStreamer/gst-rtsp-server

22 https://www.videolan.org/vlc/

Table 1 P1435-LE and Cowrie reproducibility

Port/ Service	State	Version/Notes	Cowrie Reproducibility
21 FTP	Open	Axis P1435-LE network Camera ftpd 9.80.1 (2020)	–
22 SSH	Closed	OpenSSH 7.9 (protocol 2.0)	Configurable banner
23 TELNET	Closed	Enabled via *http://IP/ admin-bin/editcgi.cgi? file=etc/inittab*	Configurable banner
25 SMTP	Closed	Send images, notifications, video clip	Forward SMTP connections to SMTP Honeypot
80 HTTP	Open	GUI, Send images, notifications, video clip	–
443 SSL	Open	Apache/2.4.41 (Unix) OpenSSL 1.1.1d	TBA
554 RTSP	Closed	GStreamer rtsp	VLC
49152 ONVIF	Closed	upnp Portable SDK for Upnp devices 1.6.22 (Linux 4.9.206-axis5; Upnp 1.0)	–

8. Enable the new service on system reboot:

```
systemctl enable axisspoof.
    service
```

9. Check the status of the new service upon reboot:

```
systemctl status axisspoof.
    service
```

During the process of getting the live stream to work, we faced some technical issues. First, installing VLC using Snap (as suggested on the VLC website) installs VLC in another directory rather than the default program directory, which should be avoided. By default, VLC does not have root permissions, and therefore it will refuse to output the video stream on port 554; instead, it outputs it on port 8544. To overcome this, VLC requires root permission for all the ports lower than 1024 [17].

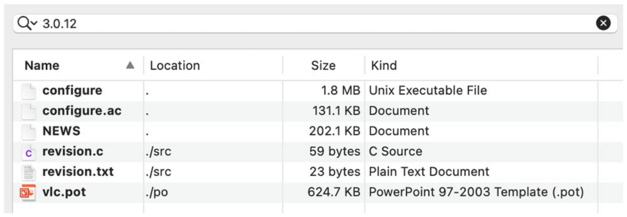
With the live stream successfully working, upon system boot and remote users being able to connect on port 554, another issue occurred. Upon performing a Nmap scan of the Ubuntu 18.04 testing environment, it was evident that Nmap was identifying "VLC rtsp 3.0.8" as the service version running on port 554 (see Listing 2).

```
PORT     STATE SERVICE VERSION
554/tcp  open  rtsp    VLC rtspd
    3.0.8
|_rtsp-methods: DESCRIBE,SETUP,
    TEARDOWN,PLAY,PAUSE,
    GET_PARAMETER
MAC Address: 00:0C:29:75:62:A1 (
    VMware)

Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel
    :4 cpe:/o:linux:linux_kernel:5
```

```
OS details: Linux 4.15--5.6
Network Distance: 1 hop
```

Listing 2 Nmap scan results of VLC implemented Axis Spoof Livefeed



3.0.12

Name	Location	Size	Kind
configure	.	1.8 MB	Unix Executable File
configure.ac	.	131.1 KB	Document
NEWS	.	202.1 KB	Document
revision.c	./src	59 bytes	C Source
revision.txt	./src	23 bytes	Plain Text Document
vlc.pot	./po	624.7 KB	PowerPoint 97-2003 Template (.pot)

Fig. 1 Files containing the VLC version string

To have VLC report to Nmap something else other than VLC when using the `-sV` or `-A` flags for port scanning, we developed a script that launches VLC as root, points to a video file, and opens an RTSP stream on the host. To do this, the VLC source code was downloaded and modified by finding all references to the VLC version string, i.e., `3.0.12` in the latest release source code at the time of writing,[23] which appears in six files (see Figure 1).

However, only three of these files need to have `3.0.12` replaced. These are `configure` (lines 592, 593, and 5227), `configure.ac` (line 5), and `vlc.pot` (line 9).

To compile, we followed the steps from https://wiki.videolan.org/UnixCompile/. However, before compiling, source code dependencies are needed within Ubuntu [13]. Additionally, because the files have been modified, before executing `./configure`, we executed `autoreconf -f -i`, which resolves any potential errors [20].

With VLC now complied and installed, VLC was set to host an RTSP stream, and accessed it via the host. While the stream was playing on the host, an Nmap scan was performed with flags `-sV` and `-A`, respectively. Note that the actual

[23] http://get.videolan.org/vlc/3.0.12/vlc-3.0.12.tar.xz

version string was replaced. However, the service was still detected correctly as `VLC rtspd` (see Listing 3).

```
PORT     STATE SERVICE VERSION
554/tcp open  rtsp    VLC rtspd X
    .Y.Z
|_rtsp-methods: DESCRIBE,SETUP,
    TEARDOWN,PLAY,PAUSE,
    GET_PARAMETER
MAC Address: 00:0C:29:75:62:A1 (
    VMware)
```

Listing 3 Nmap scan results detected the change in version, however still `VLC rtspd` as the running service

This is a major problem for the believability aspect, because it would give away the appearance of the P1435-LE camera, and an attacker may raise questions as to why an IP camera would have VLC running on it.

To determine from where the string `VLC rtspd` was pulled from, the contents of the VLC source code was searched, but no reference was found to such a string. Further research lead to Nmap's `nmap-service-probes` file, which contains a list of regex-based search patterns formatted similarly to the Perl syntax to identify the service running on the identified ports.

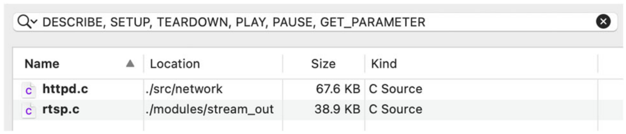
While this file could not be located on the local host, reviewing the Nmap source code online led to the `nmap-service-probes` file.[24] Doing a search for the string `VLC rtspd` within this file revealed the regular expression string used to identify the service (see Listing 4).

```
match rtsp m|^RTSP/1\.0 200 OK\r\
    nServer: VLC/([\w._-]+)\r\
    nContent-Length: 0\r\nPublic:
    DESCRIBE,SETUP,TEARDOWN,PLAY,
    PAUSE,GET_PARAMETER\r\n\r\n| p
    /VLC rtspd/ v/$1/ cpe:/a:
    videolan:vlc_media_player:$1/
```

Listing 4 Nmap's `VLC rtsp` match command

The caveat is that the string Nmap outputs describing the identify of the service is hard-coded within Nmap, rather than the VLC source code, as identified from the above Perl search pattern `p/VLC rtspd/`, is essentially `p/service/`.

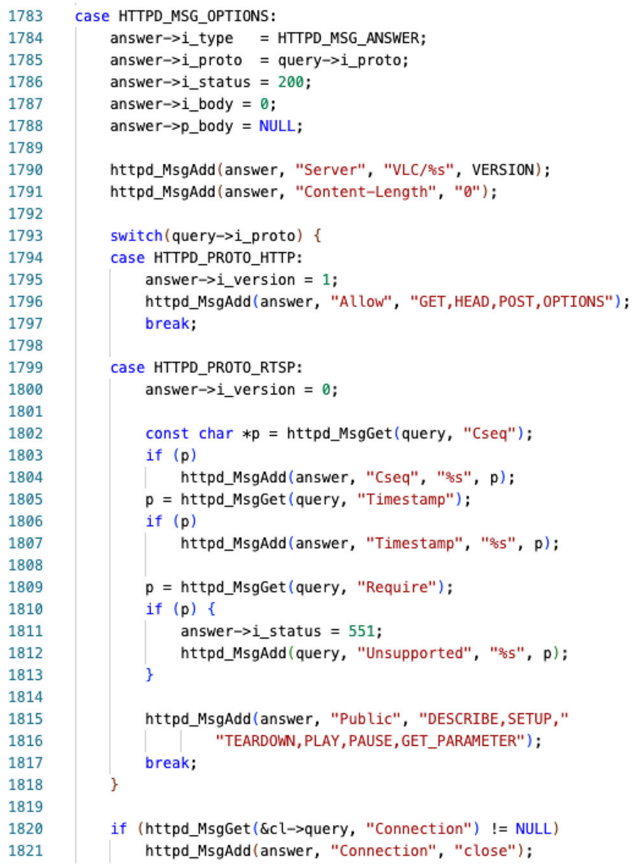
Next, it was attempted to find within the VLC source code what the `match` command is looking for so that it can be modified appropriately. We searched the VLC source code contents for the specific string `DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET_PARAMETER`, as it matches specifically what Nmap intends to find, which only appears in two files, as shown in Figure 2.



DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET_PARAMETER

Name	Location	Size	Kind
httpd.c	./src/network	67.6 KB	C Source
rtsp.c	./modules/stream_out	38.9 KB	C Source

Fig. 2 Files containing the string Nmap is searching for



```
1783  case HTTPD_MSG_OPTIONS:
1784      answer->i_type   = HTTPD_MSG_ANSWER;
1785      answer->i_proto  = query->i_proto;
1786      answer->i_status = 200;
1787      answer->i_body = 0;
1788      answer->p_body = NULL;
1789
1790      httpd_MsgAdd(answer, "Server", "VLC/%s", VERSION);
1791      httpd_MsgAdd(answer, "Content-Length", "0");
1792
1793      switch(query->i_proto) {
1794      case HTTPD_PROTO_HTTP:
1795          answer->i_version = 1;
1796          httpd_MsgAdd(answer, "Allow", "GET,HEAD,POST,OPTIONS");
1797          break;
1798
1799      case HTTPD_PROTO_RTSP:
1800          answer->i_version = 0;
1801
1802          const char *p = httpd_MsgGet(query, "Cseq");
1803          if (p)
1804              httpd_MsgAdd(answer, "Cseq", "%s", p);
1805          p = httpd_MsgGet(query, "Timestamp");
1806          if (p)
1807              httpd_MsgAdd(answer, "Timestamp", "%s", p);
1808
1809          p = httpd_MsgGet(query, "Require");
1810          if (p) {
1811              answer->i_status = 551;
1812              httpd_MsgAdd(query, "Unsupported", "%s", p);
1813          }
1814
1815          httpd_MsgAdd(answer, "Public", "DESCRIBE,SETUP,"
1816                  "TEARDOWN,PLAY,PAUSE,GET_PARAMETER");
1817          break;
1818      }
1819
1820      if (httpd_MsgGet(&cl->query, "Connection") != NULL)
1821          httpd_MsgAdd(answer, "Connection", "close");
```

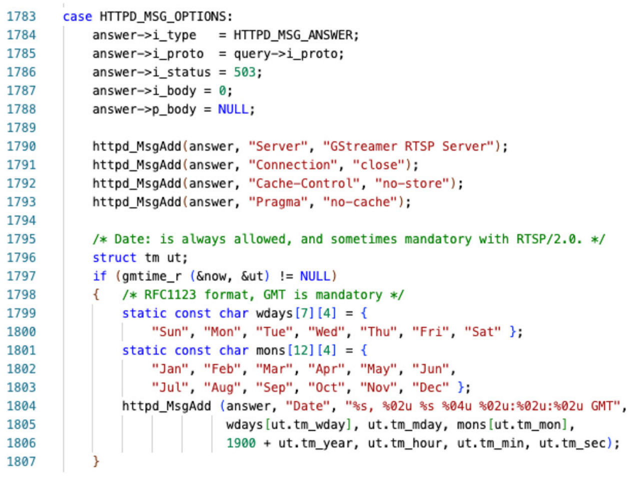
Fig. 3 Location of the `VLC rtspd match` Perl search pattern within VLC's source code

However, only the `httpd.c` requires modification. We were able to locate with probable cause that the `VLC rtspd` strings Nmap is detecting for are located between lines 1738–1821 of this file, as shown in Figure 3. It is highly likely that this is the only source which outputs the required parameters needed by the `match` command within the VLC source code.

In order to attempt to mimic GStreamer's Nmap output, it's `match` command was also identified within the `nmap-service-probes` file by doing a search for the string `GStreamer rtsp` (see Listing 5).

```
match rtsp m|^HTTP/1\.0 503
    Service Unavailable\r\nServer:
     GStreamer RTSP Server\r\
    nConnection: close\r\nCache-
    Control: no-store\r\nPragma:
```

24 https://svn.nmap.org/nmap-releases/nmap-7.90/nmap-service-probes



```
1783  case HTTPD_MSG_OPTIONS:
1784      answer->i_type   = HTTPD_MSG_ANSWER;
1785      answer->i_proto  = query->i_proto;
1786      answer->i_status = 503;
1787      answer->i_body = 0;
1788      answer->p_body = NULL;
1789
1790      httpd_MsgAdd(answer, "Server", "GStreamer RTSP Server");
1791      httpd_MsgAdd(answer, "Connection", "close");
1792      httpd_MsgAdd(answer, "Cache-Control", "no-store");
1793      httpd_MsgAdd(answer, "Pragma", "no-cache");
1794
1795      /* Date: is always allowed, and sometimes mandatory with RTSP/2.0. */
1796      struct tm ut;
1797      if (gmtime_r (&now, &ut) != NULL)
1798      {   /* RFC1123 format, GMT is mandatory */
1799          static const char wdays[7][4] = {
1800              "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
1801          static const char mons[12][4] = {
1802              "Jan", "Feb", "Mar", "Apr", "May", "Jun",
1803              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
1804          httpd_MsgAdd (answer, "Date", "%s, %02u %s %04u %02u:%02u:%02u GMT",
1805                        wdays[ut.tm_wday], ut.tm_mday, mons[ut.tm_mon],
1806                        1900 + ut.tm_year, ut.tm_hour, ut.tm_min, ut.tm_sec);
1807      }
```

Fig. 4 Modified `httpd.c` file with `GStreamer rtspd match` Perl search pattern

```
no-cache\r\nDate: .*\r\n\r\n$|
 p/GStreamer rtspd/
```

Listing 5 Nmap's `GStreamer rtsp` match command

The `httpd.c` file was then modified as follows:

- Lines 1783–1821 were replaced to contain only the required parameters and values of GStreamers Nmap detection, these being `i_status`, `Server`, `Connection`, `Cache-control`, `Pragma`, and `Date` as shown in Figure 4.
- Lines 263–287 where deleted entirely, and
- Line 234 was modified from `{ 503, "Service unavailable" },` to `{ 503, "Service Unavailable" }` — note the capital U.

After running `make uninstall` and `make clean`, and recompiling the new changes along with the replaced `3.0.12` strings as mentioned earlier, Nmap cannot detect what the service is running on the open port. Instead, it tries its best to guess what the unrecognized service is, as none of the Perl search patterns got a result. Nmap will instead create a unique fingerprint ID listing the details of its attempted detection so that the unidentified service can be officially added within the `nmap-service-probes` file and distributed in the next update. Of particular interest, when performing a scan with the `-A` flag, Nmap detects and outputs the modified changes to the source code (see Listing 6).

```
PORT    STATE SERVICE VERSION
554/tcp open  rtsp?
|_rtsp-methods: ERROR: Script
   execution failed (use -d to
   debug)
| fingerprint-strings:
|   RTSPRequest:
|     RTSP/1.255 503 Service
   Unavailable
|     Server: GStreamer RTSP
   Server
|     Connection: close
|     Cache-Control: no-store
|     Pragma: no-cache
|_    Date: Sat, 02 Sep 1978
   16:04:44 GMT
MAC Address: 00:0C:29:75:62:A1 (
   VMware)
```

Listing 6 Nmap scan results of modified VLC source code to mimic GStreamer

Given that the service is hard-coded into Nmap rather than pulling the service name from the detected open port, such is the case with the version number, this approach has challenges in terms of ease of implementation and version-dependence, given the required amount of tinkering required just to get this far. The authors believe that this should be possible to achieve with additional work.

Future work on this area includes:

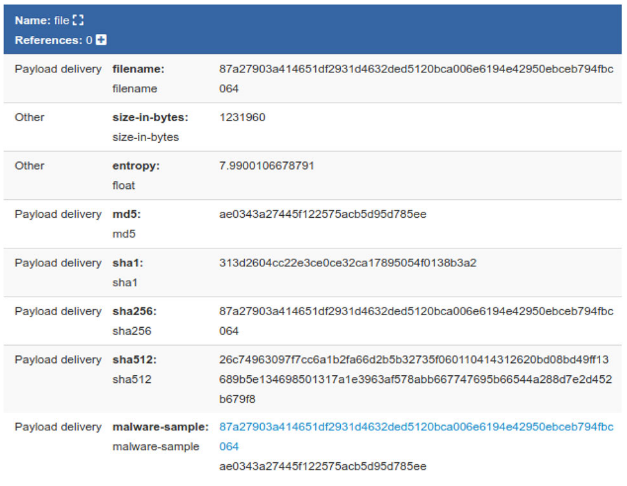
- Performing a Wireshark packet capture of Nmap's service detection probing on the modified VLC implementation,
- Further reverse engineering and modification of VLC's source code,
- Reverse engineering of GStreamer's source code, and
- Better understanding of the specifics of Nmap's fingerprint ID syntax.

5 Evaluation

Malware Information Sharing Platform (MISP)[25] is an open source utility that provides a central indicators of compromise (IoC) database that is easily accessible for both technical and non-technical users, providing information about malware and attacks, which are stored in a semistructured format [26]. The correlation features of MISP allow valuable data to be automatically provided, thereby showing the relationship between each event in the database. This data can then be exported in many different formats, including OpenIOC, plain text, and XML outputs, along with intrusion detection system (IDS) rules [26].

By using MISP with Cowrie, honeypots can automatically report findings to MISP, which can be exported in many different formats based on potential requirements. This also allows the reports of multiple honeypot instances to be col-

[25] https://www.misp-project.org



Name: file
References: 0

Payload delivery	**filename:** filename	87a27903a414651df2931d4632ded5120bca006e6194e42950ebceb794fbc064
Other	**size-in-bytes:** size-in-bytes	1231960
Other	**entropy:** float	7.9900106678791
Payload delivery	**md5:** md5	ae0343a27445f122575acb5d95d785ee
Payload delivery	**sha1:** sha1	313d2604cc22e3ce0ce32ca17895054f0138b3a2
Payload delivery	**sha256:** sha256	87a27903a414651df2931d4632ded5120bca006e6194e42950ebceb794fbc064
Payload delivery	**sha512:** sha512	26c74963097f7cc6a1b2fa66d2b5b32735f060110414312620bd08bd49ff13689b5e134698501317a1e3963af578abb667747695b66544a288d7e2d452b679f8
Payload delivery	**malware-sample:** malware-sample	87a27903a414651df2931d4632ded5120bca006e6194e42950ebceb794fbc064 ae0343a27445f122575acb5d95d785ee

Fig. 5 Attributes stored by Cowrie in MISP for the Adobe Reader DC installer file

lated into one centralized reporting system, where the number of sightings of a particular sample or file can be accurately measured from multiple systems or sessions, and the data can be correlated to provide forensically valuable statistics.

Cowrie includes a MISP module in its code that can automatically upload information about any files that are uploaded to the honeypot during an SSH session, allowing MISP to store and correlate reports about these samples instead of having to inspect the honeypot logs manually. This allows for significantly faster and easier incident reporting and data collation than attempting to parse plain-text logs manually.

By using this approach, we monitored the incidents on our proposed honeypot approach, while also providing statistics and forensically valuable information with regards to the number of sightings on a particular file or sample across numerous honeypots.

Figure 5 shows the attributes that are added into MISP by Cowrie's default MISP module when a connecting user uses the `wget` command in their SSH session to download the Adobe Reader DC installer.[26] The information includes file size, entropy, multiple hashes, and the sample itself. Note that although the sample is stored as "malware-sample", it may not always be malicious, because non-malicious data can also be uploaded to Cowrie.

By default, the hashes and the sample files are correlated, allowing MISP to link similar events that involve files with the same hashes together. However, if the exact same file is uploaded to Cowrie, it will just increment the number of sightings in MISP rather than creating a new event, which reduces duplicate reports but results in correlation charts not displaying every instance of the event.

[26] http://admdownload.adobe.com/bin/live/readerdc_en_a_install.exe

6 Conclusions

This paper presented a novel approach for honeypot-based hardening of video camera security, and presented an implementation for an Axis camera model. In particular, this approach provides mechanisms to mimic the Axis P1435-LE camera, spoof videostreams through VLC, while keeping believability, frame boundaries, and video file properties in mind. This requires taking the characteristics of the chosen device model to mimic into account to an extent that adversaries—at least initially—will not realize that they are interacting with a honeypot and not a video camera. The proprietary solutions, varying device characteristics, and software implementations all pose challenges in terms of practicality and scalability, and in particular, prevent device-specific components and streamlined development practices for future software versions and device models. This approach has been evaluated using MISP, which allows for generating reports, identifying threats and camera hacking attempts, and linking similar events.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Alsmirat, M.A., Obaidat, I., Jararweh, Y., Al-Saleh, M.: A security framework for cloud-based video surveillance system. Multimedia Tools Appl. **76**, 22787–22802 (2017). https://doi.org/10.1007/s11042-017-4488-1
2. Axis Communications: Axis introduces two new bullet-style HDTV network cameras for difficult light conditions (2015). https://www.axis.com/files/press_releases/pr_p1435e_p1435le_1512.pdf
3. Axis Communications: Acv-128401affected product list (2018). https://www.axis.com/files/sales/ACV-128401_Affected_Product_List.pdf
4. Axis Communications: AXIS P1435-LE network camera compact and fully-featured HDTV for any light condition (2020). https://www.axis.com/files/datasheet/ds_p1435le__t10054259_en_2005.pdf
5. Bäckman, M., Hagfjäll, F.: Application security for embedded systems. Master's thesis, Department of Electrical and Informa-

tion Technology, Lund University (2017), https://www.eit.lth.se/sprapport.php?uid=1032

6. Cabral, W.Z., Valli, C., Sikos, L.F., Wakeling, S.G.: Review and analysis of Cowrie artefacts and their potential to be used deceptively. In: 2019 International Conference on Computational Science and Computational Intelligence, IEEE, pp. 166–171 (2019), https://doi.org/10.1109/CSCI49370.2019.00035
7. Cabral, W.Z., Valli, C., Sikos, L.F., Wakeling, S.G.: Analysis of Conpot and its BACnet features for cyber-deception. In: Daimi, K., Arabnia, H.R., Deligiannidis, L., Hwang, M.S., Tinetti, F.G. (eds.) Advances in Security, Networks, and Internet of Things, pp. 329–339. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71017-0_23
8. Chen, Y., Lian, X., Yu, D., Lv, S., Hao, S., Ma, Y.: Exploring Shodan from the perspective of industrial control systems. IEEE Access **8**, 75359–75369 (2020). https://doi.org/10.1109/ACCESS.2020.2988691
9. Costin, A.: Security of CCTV and video surveillance systems: threats, vulnerabilities, attacks, and mitigations. In: Proceedings of the 6th International Workshop on Trustworthy Embedded Devices, ACM, New York, pp. 45–54 (2016), https://doi.org/10.1145/2995289.2995290
10. Cusack, B., Tian, Z.: Evaluating IP surveillance camera vulnerabilities. In: Valli C (ed.) Australian Information Security Management Conference, Springer, Heidelberg, pp. 25–32 (2017), https://doi.org/10.4225/75/5a84efba95b46
11. Dowling, S., Schukat, M., Barrett, E.: Using reinforcement learning to conceal honeypot functionality. In: Brefeld U, Curry E, Daly E, MacNamee B, Marascu A, Pinelli F, Berlingerio M, Hurley N (eds) Machine Learning and Knowledge Discovery in Databases, Springer, Cham, pp. 341–355 (2019), https://doi.org/10.1007/978-3-030-10997-4_21
12. Egashira, T., Meng, L., Tomiyama, H.: A home security camera system based on cloud and SNS. In: Chiplunkar NN, Fukao T (eds) Advances in Artificial Intelligence and Data Engineering, Springer, Singapore, pp. 1375–1381 (2020), https://doi.org/10.1007/978-981-15-3514-7_103
13. Exchange, S.: Error : you must put some 'source' URIs in your sources.list (2015). https://askubuntu.com/questions/496549/error-you-must-put-some-source-uris-in-your-sources-list
14. Guarnizo, J., Tambe, A., Bhunia, S.S., Ochoa, M., Tippenhauer, N.O., Shabtai, A., Elovici, Y.: SIPHON: towards scalable high-interaction physical honeypots. In: Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, ACM, New York, pp. 57–68 (2017), https://doi.org/10.1145/3055186.3055192
15. Guri, M., Bykhovsky, D.: aIR-Jumper: covert air-gap exfiltration/infiltration via security cameras & infrared (IR). Comput. Secur. **82**, 15–29 (2018). https://doi.org/10.1016/j.cose.2018.11.004
16. Kaliappan, V., Yu, S., Soundararajan, R., Jeon, S., Min, D., Choi, E.: High-secured data communication for cloud enabled secure docker image sharing technique using blockchain-based homomorphic encryption. Energies **15**(15), 89 (2022). https://doi.org/10.3390/en15155544
17. Kili, A.: How to install and run VLC Media Player as root in Linux (2017). https://www.tecmint.com/run-vlc-media-player-as-root-in-linux/
18. Liu, J., Gu, Y., Kamijo, S.: Customer pose estimation using orientational spatio-temporal network from surveillance camera. Multimedia Syst. **24**, 439–457 (2018). https://doi.org/10.1007/s00530-017-0570-9
19. Luo, T., Xu, Z., Jin, X., Jia, Y., Ouyang, X.: IoTCandyJar: towards an intelligent-interaction honeypot for IoT devices. In: Black Hat USA 2017 (2017), https://www.blackhat.com/docs/us-17/thursday/us-17-Luo-Iotcandyjar-Towards-An-Intelligent-Interaction-Honeypot-For-IoT-Devices-wp.pdf
20. Overflow, S.: How to overcome "aclocal-1.15' is missing on your system" warning? (2016), https://stackoverflow.com/questions/33278928/how-to-overcome-aclocal-1-15-is-missing-on-your-system-warning/33279062
21. Peles, O.: Vdoo discovers significant vulnerabilities in Axis cameras (2018). https://www.vdoo.com/blog/vdoo-discovers-significant-vulnerabilities-in-axis-cameras
22. Rowe, N.C., Duong, B.T., Custy, E.J.: Fake honeypots: a defensive tactic for cyberspace. In: Proceedings of the 2006 IEEE Information Assurance Workshop, IEEE, pp. 223–230 (2006), https://doi.org/10.1109/IAW.2006.1652099
23. Saini, D.K., Ahir, D., Ganatra, A.: Techniques and challenges in building intelligent systems: Anomaly detection in camera surveillance. In: Satapathy SC, Das S (eds) Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems, vol. 2, pp. 11–21, Springer, Cham (2016), https://doi.org/10.1007/978-3-319-30927-9_2
24. Singh, D.K., Kushwaha, D.S.: Tracking movements of humans in a real-time surveillance scene. In: Pant M, Deep K, Bansal JC, Nagar A, Das KN (eds) Proceedings of Fifth International Conference on Soft Computing for Problem Solving, Springer, Singapore, pp. 491–500 (2016), https://doi.org/10.1007/978-981-10-0451-3_45
25. Sitara, K., Mehtre, B.M.: Automated camera sabotage detection for enhancing video surveillance systems. Multimedia Tools Appl. **78**, 5819–5841 (2019). https://doi.org/10.1007/s11042-018-6165-4
26. Skopik, F., Settanni, G., Fiedler, R.: A problem shared is a problem halved: a survey on the dimensions of collective cyber defense through security information sharing. Comput. Secur. **60**, 154–176 (2016). https://doi.org/10.1016/j.cose.2016.04.003
27. Tambe, A., Aung, Y.L., Sridharan, R., Ochoa, M., Tippenhauer, N.O., Shabtai, A., Elovici, Y.: Detection of threats to IoT devices using scalable VPN-forwarded honeypots. In: Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, ACM, New York, pp. 85–96 (2019), https://doi.org/10.1145/3292006.3300024
28. Tekeoglu, A., Tosun, A.S.: Investigating security and privacy of a cloud-based wireless IP camera: NetCam. In: Proceedings of the 24th International Conference on Computer Communication and Networks, IEEE (201x), https://doi.org/10.1109/ICCCN.2015.7288421
29. Tomar, A., Mishra, P., Bisht, R., Kumar, P.: A step towards generation of DoS/DDoS attacks dataset for docker-centric computing. Int. J. Math. Eng. Manag. Sci. **7**(1), 81–91 (2022). https://doi.org/10.33889/IJMEMS.2022.7.1.006
30. Vempati, J., Dantu, R., Thompson, M.: Uninterrupted video surveillance in the face of an attack. In: Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, IEEE, pp. 843–848 (2018), https://doi.org/10.1109/TrustCom/BigDataSE.2018.00121
31. Wu L, Lovell BC, Wang Y (2019) Deep learning in person re-identification for cyber-physical surveillance systems. In: Alazab M, Tang M (eds) Deep Learning Applications for Cyber Security, Springer, Cham, pp 45–72, https://doi.org/10.1007/978-3-030-13057-2_3