SweetCam: an IP Camera Honeypot

Zhao, Zetong; Srinivasa, Shreyas; Vasilomanolakis, Emmanouil

Published in:
Proceedings of the 4th Workshop on CPS & IoT Security and Privacy (CPSIoTSec'23)

Link to article, DOI:
10.1145/3605758.3623495

Publication date:
2023

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Zhao, Z., Srinivasa, S., & Vasilomanolakis, E. (2023). SweetCam: an IP Camera Honeypot. In Proceedings of the 4th Workshop on CPS & IoT Security and Privacy (CPSIoTSec'23) (pp. 75–81). ACM. https://doi.org/10.1145/3605758.3623495

SweetCam: an IP Camera Honeypot

Zetong Zhao
s223057@student.dtu.dk
Technical University of Denmark
Kongens Lyngby, Denmark

Shreyas Srinivasa
shsr@es.aau.dk
Aalborg University
Copenhagen, Denmark

Emmanouil Vasilomanolakis
emmva@dtu.dk
Technical University of Denmark
Kongens Lyngby, Denmark

ABSTRACT

The utilization of the Internet of Things (IoT) as an attack surface is nowadays a fact. Taking IP cameras as a use-case, they have been targeted to a great extent mainly due to the absence of authentication, the utilization of weak, in terms of security, protocols, and their high availability. To cope with the current situation and study the current state of attacks against IP cameras we propose the use of cyber-deception and in particular honeypots. Honeypots can provide useful insights into current attack campaigns, and they can divert attackers' attention away from the actual targets.

In this paper, we propose an open-source medium interaction IP camera honeypot that requires minimal settings while supporting a modular architecture for adding new camera models. The honeypot, namely *SweetCam*, supports the emulation of SSH, RTSP and HTTP. Furthermore, it creates a web-service (HTTP) that depicts an IP camera interface with a login page and the emulation of a camera interface using user-specified 360-degree video streams and images. We deploy instances of the honeypot in different geographical locations, for a period of 3 weeks, and receive a total of 5,780, 1,402 and 218,344 attacks on HTTP, RTSP and SSH services respectively; from 5,924 unique IPs. Lastly, we further analyze the attacks, and identify common Internet scanners (e.g., Shodan) among the services that have contacted the honeypots.

CCS CONCEPTS

• **Security and privacy** → **Network security**.

KEYWORDS

Honeypot, IP camera, Network Deception, Network Security, IoT Security

ACM Reference Format:
Zetong Zhao, Shreyas Srinivasa, and Emmanouil Vasilomanolakis. 2023. SweetCam: an IP Camera Honeypot. In *Proceedings of the 4th Workshop on CPS&IoT Security and Privacy (CPSIoTSec '23), November 26, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3605758.3623495

1 INTRODUCTION

The Internet of Things (IoT) is nowadays considered a very common target of cyber attacks. In particular, many adversaries first compromise a large number of IoT devices and subsequently use them to perform attacks [17] (see for example the case of the Mirai botnet [1]). Among the targeted IoT devices, recent research suggests that IP cameras face numerous threats stemming from their inherent vulnerabilities [11]. These vulnerabilities may arise from outdated or faulty security modules, inadequate patch management, insufficient access control measures, weak physical security, etc. Furthermore, IP cameras offer a low hanging fruit due to their constant connectivity and low IP churn along with the good enough hardware capabilities and poor security.

To understand the attack landscape and the trends in IP cameras, a valuable approach is the use of honeypots. They are widely used to identify active threats, gather evidence of attacks, and engage, and mislead attackers by emulating specific functionalities of systems or devices [20]. In addition, a honeypot can also record attackers' techniques when they perform lateral movement within a network. Honeypot gathered data can be used to analyze attacks, find potential zero-day vulnerabilities, etc. [19, 23]. There have been two main classes of IP camera honeypots: one is building honeypots based on real IP devices using a port forwarding approach; and the other one is building honeypots using ordinary prerecorded videos. However, both these classes of honeypots experience certain drawbacks. For the first approach, since a real device is required, it adds cost and complexity to both setting up the honeypot but also monitor it. As for the second approach, it lacks a key feature of the IP camera, which is to move and change the camera view based on the attackers' control. In this case, the drawback lies in the realisticness of the deception.

In this paper, we propose *SweetCam* an open-source medium interaction honeypot for IP cameras, that is lightweight, does not require real devices, and allows attackers to interact with it in a realistic manner. Our honeypot lures attackers by offering a web-UI that accepts weak username-password combinations and subsequently presents the adversary with a realistic camera feed. To complement the deceptive nature of *SweetCam* we natively support HTTP, and RTSP, while SSH is also emulated via the utilization and integration of the state-of-the-art honeypot Cowrie.

The remainder of this paper is as follows. In Section 2, we discuss the related work and in Section 3 we present *SweetCam*'s design and architecture. Moreover, in Section 4 we present the results from a three-week deployment of the honeypot. Lastly, Section 5 concludes this paper.

2 RELATED WORK

Honeypots are deception-based systems used to allure attackers by providing them with fake targets and gathering the attack traces for defense study. With years of development, many honeypots targeting various applications and products have emerged. The Honeynet Project provides many open-source honeypots that are based on simulating either specific services or devices [15]. For example, Cowrie [2] is a medium-interaction honeypot intending to record possible attacks launched on SSH and Telnet. Cowrie often works as a component of a bigger honeypot like TPot [14], which pattern is also adopted in this project. The Glastopf honeypot [10] is designed for web applications and is a low-interaction honeypot that emulates a vulnerable web server hosting many web pages with known vulnerabilities [9].

In recent years, with the fast development of IoT, the respective honeypots gained increasing attention [13, 18, 22]. These honeypots are developed to protect IoT devices or study the attack landscape. For example, Wang et al. proposed IoTCMAL, a hybrid IoT honeypot framework for capturing comprehensive malicious samples aimed at IoT devices, based on the observations that there are two types of services in IoT devices that are easily exploited by attackers, namely the weak authentication services (e.g., SSH/Telnet) and the exploited services using command injection [25]. Luo et al. proposed an approach to build IoT honeypots automatically and intelligently with IoTCandyJar [8]. The honeypot leverages machine learning and publicly available IoT devices (including IP cameras) on the Internet to generate potential responses.

Within all IoT devices, the IoT camera (IP camera) is a vital class and is widely used in surveillance ecosystems. Targeting IoT cameras, several honeypots have been developed to enhance the overall security of IoT camera systems. U-POT, an IoT honeypot framework for UPnP (Universal Plug and Play) protocol, was introduced by Hakim et al. [6]. The honeypot simulates the Universal Plug and Play (UPnP) protocol, which many IoT devices use, including smart switches, bulbs, and IP cameras. U-POT can utilize device description files to automate honeypots and generate fake responses. Vishwakarma et al. managed to combine the IoT honeypots with machine learning technologies [24]. They obtained the attack data from IoT honeypots and used it as training data to train the machine learning model to study the newer emerging variants of IoT malware and combat the Zero-Day attacks. However, these kinds of honeypots can only handle attacks launched by scripts and are unable to engage human attackers for a prolonged period due to the lack of video streaming functionality.

Facing this challenge, SIPHON, a high-interaction IoT camera honeypot architecture, was proposed by Guarnizo et al. [4]. This architecture deploys honeypots on the cloud and uses traffic forwarding techniques to redirect attacker commands to real IoT cameras in their laboratory to give attackers the feel of interacting with real IP cameras. Similarly, IoTCMAL, a hybrid IoT honeypot framework proposed by Wang et al. uses real IP cameras to back up the deployment of honeypots using traffic forwarding [25]. Nevertheless, this approach has limited scalability due to the requirement of actual physical devices with extra network bandwidth and is expensive.

In another approach, Tabari et al. proposed a multi-phased approach to build a honeypot ecosystem and built a low-interaction honeypot for IoT cameras named HoneyCamera which mimics the D-Link IP camera and can continuously present recorded videos to attackers to engage them [26]. In this way, the authors gathered many attacking traces exploiting existing D-Link vulnerabilities. Though recorded videos are easier to obtain and use, they are not very convincing for human attackers since the attacker can not interact with the provided video image.

To allow the honeypot that uses the recorded video to interact with the attacker, Guan et al. proposed a way to make the video image to be able to rotate and zoom in/out based on user control by using 360-degree video and building a honeypot named HoneyCam [3]. However, their solution requires an extra video processing unit that presents different video images based on the control signal sent by the attacker. Moreover, the used 360-degree video requires special equipment and is not easy to make.

Table 1 provides a comparison of the state of the art and SweetCam. For comparison, we classify the honeypots in Table 1 into four classes.

- The first class includes U-POT and IoTCandyJar, which only partly mimic the IP camera, they lack many vital features (like one camera video stream) and can be easily fingerprinted.
- The second class includes SIPHON and IoTCMal, since they are using real devices as the backup of the honeypots, they have a relatively comprehensive simulation of real IP cameras. However, just as the aforementioned, the drawbacks also come from the use of real devices, they add setting cost and complexity, and sometimes are not suitable.
- The third type only includes HoneyCamera, which uses prerecorded video instead of real devices. Since there is no special process for the used videos, it does not support any attacker interaction.
- The last class includes HoneyCam and *SweetCam*, which use 360-degree videos as interaction to the attacker. However, HoneyCam requires a special *video processing unit*, which processes the 360-degree videos based on attackers' instructions and returns proper video images. When compared to *SweetCam*, this approach is more complicated, and expensive. Guan et al. hosted the front-end application of HoneyCam in the cloud and prepared a server to host the back-end application together with the video processing unit [3]. On SweetCam, since the video image transformation function is integrated into the front-end application there is no need to set up a separate server, making it easier to set up and low costs. Besides, as mentioned in Section 3.1, most of the time 360-degree images suffice the purpose. Moreover, *SweetCam* supports the use of 360-degree images while HoneyCam does not. In addition, *SweetCam* supports SSH through the integration of the Cowrie honeypot.

Based on these limitations, we investigate how to provide a user interaction function without using an additional processing unit and use more easily obtained 360-degree images.

3 DESIGNING AN IP CAMERA HONEYPOT

In this section, we present the design of *SweetCam*, a medium interaction IP camera honeypot. *SweetCam* has two types of users. The first one is the attacker, which refers to those expected to attack the honeypot; the other one is the administrator, which refers to those who deploy the honeypots and gather attack data. The *SweetCam* honeypot is composed of three parts:

- The web service. This service is responsible for serving the front-end web pages for the attackers to log in and view the camera images. And also works to provide back-end services

Name	Web UI	Camera Video Stream	Does not require real device	RTSP service	SSH protocol	User Interaction
U-POT [6]	✗	✗	✓	✗	✗	✗
IoTCandyJar [8]	✗	✗	✓	✗	✗	✗
SIPHON [4]	✓	✓	✗	✓	✗	✓
IoTCMal [25]	✓	✓	✗	✗	✓	✓
HoneyCamera [26]	✓	✓	✓	✗	✗	✗
HoneyCam [3]	✓	✓	✓	✓	✗	✓
SweetCam	✓	✓	✓	✓	✓	✓

Table 1: Comparison between SweetCam and the state of the art

that support the front-end service together with a set of APIs for the administrator to manage the application.

- The RTSP service. This service enables the attackers to view the camera video through the RTSP protocol.
- The SSH service. This service provides a fake SSH function for the attackers and records their used commands.

SweetCam is offered as an open-source project [1] and the architecture is shown in Figure 1. Here we can see that each of the three components should have a proper logging system for recording the traces of attackers for further study investigation.

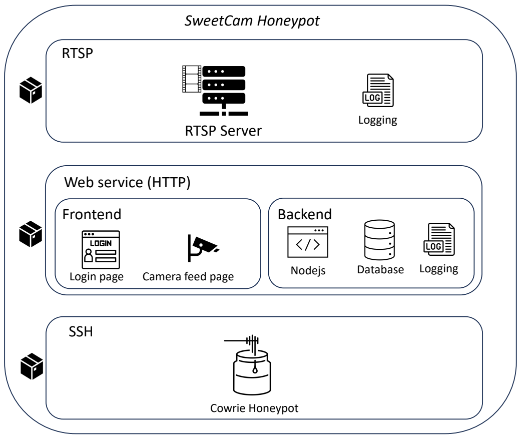

Figure 1: The architecture of *SweetCam*

3.1 Web Service

The web service consists of the front-end service and the back-end service. We describe each as follows:

3.1.1 Front-end. The front-end contains three pages. One is used for user login, and the other two are used for displaying the camera images. The difference between the two camera images displaying pages is the used media, one uses 360-degree video, and the other one uses 360-degree image. The reason that the application provides the function to choose between 360-degree video and 360-degree image is that 360-degree image is much easier to make compared with 360-degree videos. The 360-degree picture is captured with a smartphone using the built-in panoramic shot mode. As for 360-degree videos, they are generally shot with a panoramic camera. However, these kinds of cameras are expensive and are more suitable for dynamic scenarios. Using 360-degree images is suitable for static scenes like a room, as most of the time, there are no changes over time. In this case, this is no need to buy specific devices and make 360-degree videos. The camera page is shown in Figure 2.

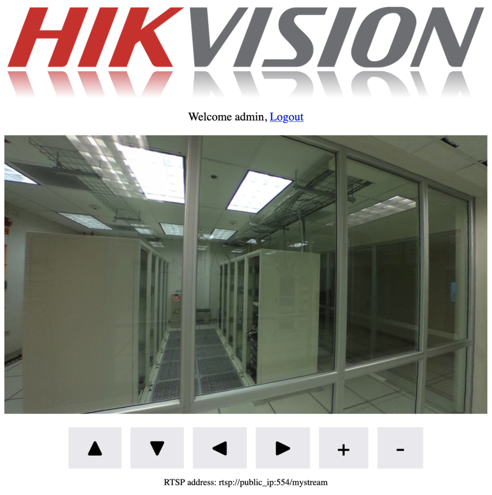

Figure 2: The SweetCam camera UI

Here, six control buttons are provided together with the camera page. The attacker can use the six control buttons to rotate the image horizontally or vertically and zoom in/out the image just like controlling a real IP camera. This function is built using the *Three.js* Javascript library. Three.js is a cross-browser JavaScript library used to create and display animated 3D computer graphics in a web browser using WebGL.

Three.js manages various objects in a 3D scene through the "Scene" object. The Scene object includes a three-dimensional coordinate system where objects are positioned at different locations. A camera is placed at a specific position within the scene to observe the objects within the Scene. The camera can capture a two-dimensional view of the observed objects, which can be rendered using the "Renderer" object. This process describes how Three.js creates a 3D scene and renders it into a 2D representation. 360-degree videos/images are videos/images that can be viewed from any direction. Based on this feature, we can put a sphere into the aforementioned 3D scene, pasting the 360-degree videos/images to the sphere, and then put a camera within it. Thereafter, we can

[1]https://github.com/Agachily/sweetcam

change the viewed image by changing the location of the camera. Note that Three.js provides lots of cameras, and here the used camera is "PerspectiveCamera"[2]. The "PerspectiveCamera" observes objects according to the perspective principle, presenting views just like being observed by human eyes.

When the attacker clicks the control buttons, corresponding parameters will be updated, resulting in the camera view being changed accordingly and giving the attacker the illusion that they are manipulating a real device. Besides, when the attacker clicks that button, the front-end page will not respond immediately since a deliberate timeout has been set before it updates the camera view. When interacting with a real device, a timeout is inevitable since the device needs time to process the single and make a response as well as the possible network transmission delay. The front-end page is written using the Pug engine [7], a template engine suitable for Node.js. Currently, *SweetCam* has a primary simulation of the Hikvision camera as shown in Figure 2. However, *SweetCam* can be easily extended, by modifying the front-end and adding more device-related elements to make it look realistic via the Pug engine. As long as the Three.js code holds intact, the interaction function is not influenced.

3.1.2 Back-end. The back-end is used to support the front-end service and provide a set of APIs for the administrator to customize the appearance and behavior of the honeypot. They are developed using Node.js and the Express framework, together with the MySQL database for data storage. The content of the MySQL database is rather simple, only contains two tables storing the credentials of the fake users (weak passwords for attackers to guess) and the administrators.

For the front-end service, the back-end supports the attackers' login and serves the camera page once the login succeeds. For the administration APIs, administrators can use them to change the displayed banner, update the displayed media, configure the size of the camera page etc. This makes the honeypot to be configurable, allowing the attackers to mimic different types of IP cameras.

3.2 RTSP Service

The RTSP service is used for the attackers to view the video through the RTSP protocol. This service is built using MediaMTX[3] and FFmpeg [21]. MediaMTX is a ready-to-use and zero-dependency server and proxy that allows users to publish, read and proxy live video and audio streams. It supports several protocols including RTSP. However, MediaMTX can only serve the video to users, and the video should be pushed to the MediaMTX server in advance. That is why FFmpeg is used; it is a complete, cross-platform solution to record, convert and stream audio and video. Here, it is used to push media to the MediaMTX server. For example, once the MediaMTX is started, by default, it provides service at the address rtsp://localhost:8554/mystream, once the video is pushed to that address using FFmpeg, it can be viewed through that address using specific tools like the VLC player[4].

[2]https://threejs.org/docs/#api/en/cameras/PerspectiveCamera
[3]https://github.com/bluenviron/mediamtx
[4]https://www.videolan.org/vlc/

3.3 SSH honeypot service

Honeypots should be realistic enough to gain attackers' interest, therefore, besides the service related to IP cameras, we can provide some other services that attackers value. IP cameras often support SSH for administrative purposes. Not surprisingly, attackers target SSH to control the IP camera for vicious usages, like being part of the botnet [16]. For example, a botnet named RapperBot [12] targeting Linux devices attempts to hack these devices by launching brute-forcing attacks on weak or default credentials.

Based on these considerations, we integrate an SSH honeypot within our honeypot application and record how attackers attempt to utilize the IP camera through SSH. Here, we adapt an existing SSH honeypot, Cowrie[5], due to its high-quality emulation, the ability that it offers to the attacker for downloading their code (e.g., via *wget*), and the fact that it is open source and highly regarded in the community [2].

3.4 Containerization

As mentioned before, the SweetCam honeypot is composed of three sub-services. Each of the services will run as a separate Docker Container. Among them, the web services and RTSP service are built based on the customized Dockerfile. As for the Cowrie service, since it has already been published to Docker Hub as an off-the-shelf container, it can be used directly, without the necessity to define the corresponding Dockerfile and build it separately. The whole SweetCam application is defined using a docker-compose.yml configuration file, it defines how the containers should be launched and cooperate, beside the above three containers, it also defined a MySQL container to support the web service. Based on this file, the SweetCam application can be easily started using Docker Compose with a single command, making it easy to be deployed either on the cloud or on physical devices (e.g., a Raspberry Pi).

4 EVALUATION

In this section, we present the experimental setup, analysis of the results from a three weeks experiment, and some insights from the attacks. Lastly, we provide a qualitative comparison between IP camera honeypots.

4.1 Experimental setup

To evaluate SweetCam, it was deployed on the Azure cloud in Brazil, South Africa, and Japan respectively. The honeypot was deployed on virtual machines having 2GB memory and one VCPU core with Ubuntu 18.04 as the operating system. To mimic the behavior of a real IoT camera, ports 554 (RTSP), 80 (HTTP), and 22 (SSH) are open and exposed to the Internet. The RTSP service enables attackers to view the video. The HTTP provides a web service and the SSH service provides access to the fake shell simulated using the Cowrie honeypot. All the logs are stored at individual virtual machine instances in *JSON* format to facilitate further analysis.

4.2 Attack data analysis

We deployed the SweetCam honeypot on the three locations mentioned above from June 5 2023 to June 27, 2023. During this time

[5]https://github.com/cowrie/cowrie

we have gathered around 706 KB logs from the web service, 222 KB logs from the RTSP service, and 689 MB data from the SSH honeypot, making a total of 1,543,402 entries as shown in Figure 3.

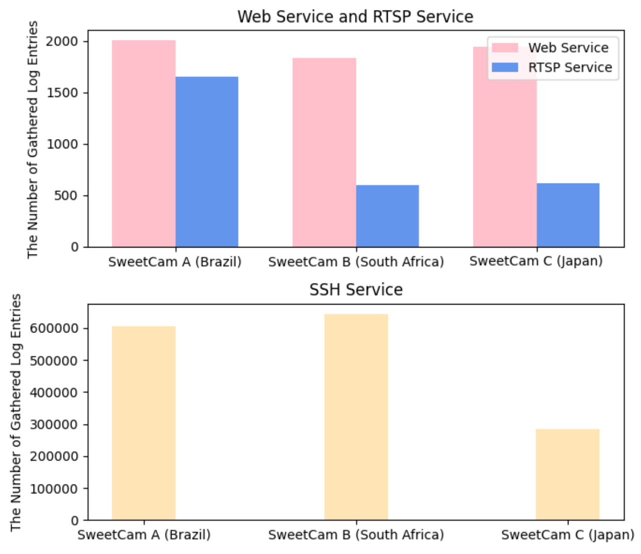

Figure 3: The number of gathered logs of three services from all honeypots

From this figure, we can see that the SSH service has drawn much more attention compared with the other two services, demonstrating that it is reasonable that we integrate an SSH honeypot into SweetCam. Figure 4 shows how the number of attack logs changes per day in the three honeypots. We can see that the honeypot deployed in Brazil and South Africa generally received more attacks compared with the one deployed in Japan.

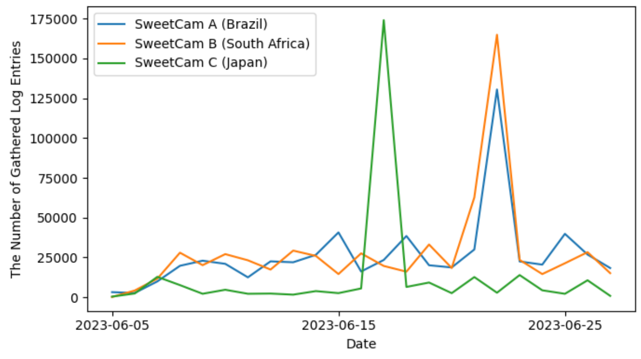

Figure 4: Changes in the number of attack log entries gathered per day in all honeypots

For the web service, most attackers did not try to log in to the web service and view the camera page. There are only 13 requests that send the non-empty body to the /login path. Among them three have the sent body "{'0x5B]': 'androxgh0st'}", made by two unique IPs. These request looks like made by the AndroxGh0st malware[6]. Among all the 5,780 lines of logs, there are 1,714 of them contains the string "androxgh0st" in the body of POST request, accounting for 30% percent of the total data. Therefore, the AndroxGh0st malware seems rather popular and active. The rest ten are all using the *'username: admin, password: admin'* combination made by 5 unique IPs. Other combinations are not observed.

For the RTSP service, most attackers are simply making requests to invalid URLs. Some typical URLs include '/', '/cgi-bin/nobody/', '/nice%20ports%2C/Tri%6Eity.txt%2ebak', etc. The request to URL '/cgi-bin/nobody/' denotes the attack on the AVTECH devices [7], targeting the *Improper Access Control* vulnerability. The presence of this vulnerability enables an external attacker to attain unauthorized entry to functionality that is typically restricted. This vulnerability arises from inadequate access controls applied to scripts housed within the "/cgi-bin/nobody" directory, such as "/cgi-bin/nobody/Machine.cgi". By sending requests to the scripts within this directory, a remote attacker who lacks authentication can exploit this flaw to retrieve sensitive information [8]. Moreover, the request to URL '/nice%20ports%2C/Tri%6Eity.txt%2ebak' denoting it is a request made by NMap. According to [5] the request for "/nice ports,/Trinity.txt.bak", comes from Nmap's service detection routine testing how a server handles escape characters within a URI. Based on the SSH attack data, we analyzed the credentials that are mostly used by attackers, the top ten of them are shown in Table 2. Upon researching the password *345gs5662d34*, we find that it is possibly the default credential for a Polycom CX600 IP telephone.

Username	Password	Times
345gs5662d34	345gs5662d34	24,956
root	3245gs5662d34	24,922
root	password	1,340
root	root	939
test	test	651
postgres	postgres	629
ansible	ansible	586
nproc	nproc	561
root	test123	548
root		543

Table 2: 10 most used credential combinations targeting SSH

Overall, there are 131,656 entries recorded in the commands that the attacker used, and among them, 7,133 entries contain *curl* commands and 4,850 entries contain *wget* contain (there is an intersection between those containing *curl* command and those containing *wget* command). The *curl* and *wget* commands are put specific attention because the attacker often uses them to download malware and execute it on the target machine. For example, Figure 5 depicts the case of a log snippet in which the attacker attempts to use *curl* to download a distributed denial-of-service (DDoS) attack script written in *Perl*.

[6]https://www.fortiguard.com/threat-signal-report/5066/androxgh0st-malware-actively-used-in-the-wild
[7]https://www.cybersecurity-help.cz/vdb/SB2016101133
[8]https://www.cybersecurity-help.cz/vulnerabilities/48219/

```
{"eventid":"cowrie.command.input","input":"uname -a;lspci | grep -i --color
'vga\\|3d\\|2d';curl -s -L http://[redacted]:8088/iposzz/dred -o /tmp/dred;perl
/tmp/dred","message":"CMD: uname -a;lspci | grep -i --color 'vga\\|3d\\|2d';curl -s -L
http://[redacted]:8088/iposzz/dred -o /tmp/dred;perl /tmp/dred","sensor":"b8a52c14b946",
"timestamp":"2023-06-14T02:33:36.002629Z","src_ip":"[redacted]",
"session":"88a533342e8b"}
```

Figure 5: Example of an attempt to download a DDoS tool using curl

Among the attacks received we observe a number of multistage attacks. Multistage attacks can be defined as attacks originating from the same source IP that sequentially target multiple protocols on the end system. Figure 6 shows the multistage attacks observed across all three simulated protocols. A total of 322 multistage attack sessions were observed and we notice that a high number of multistage attacks originated from the Web (HTTP). Note that the total count in the figure represents the sessions and not the number of attacks. The attack sessions were grouped based on a five-minute time window. We observe that the adversaries tried to deploy scripts to fetch malware and checked for other open services on the end systems. Most of the attacks observed on the SSH protocol are brute-force attacks, followed by HTTP web scraping on the HTTP and the camera stream access on the RTSP.

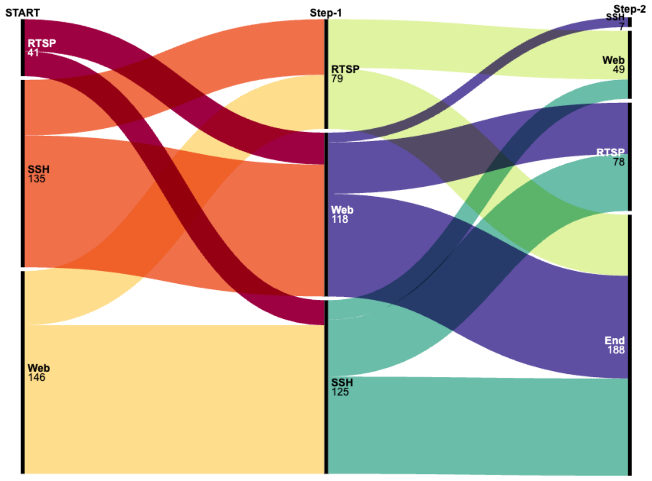

Figure 6: Multistage attacks: same attacker IP targeting multiple protocols

We filter the noise from the traffic received on the honeypot instances. Figure 7 shows the total source IPs and percentage of IPs that belong to Internet scanning services like Censys, Shodan and the other suspicious IPs. We classify the traffic from 19 known Internet scanning services and observe that the Web (HTTP) service had the highest number of noise in comparison to the SSH and the RTSP services.

4.3 Comparison between *SweetCam* and typical honeypots: HoneyCam and Siphon

Siphon is a typical honeypot that utilizes real devices to design the IP camera. This approach has many advantages. Using the port

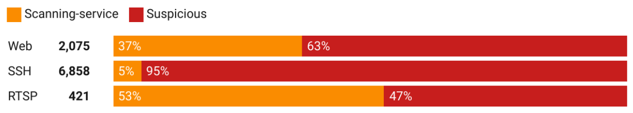

Figure 7: Noise classification and total number of unique IPs per protocol

forwarding technology, the user is basically interacting with the real device, thus they can view the real user interface and interact with the real device. The main disadvantage of such a high interaction approach is the cost to set up the honeypot because the usage of real hardware. Another drawback is that the administrator needs to ensure that the video output of the camera does not introduce privacy or other issues. In this case, it is better to replay synthetic or fake videos instead.

As for HoneyCam, it cleverly uses 360-degree videos to provide fake interaction functions to the attackers. However, HoneyCam does not process the video at front-end like our system. Instead, it has a separate unit for processing the 360-degree video based on the attacker's operation and sending the processed video back to the front-end to play. The separate processing unit also adds up costs and leads to the risk of single-point failure, especially when many honeypots are deployed. If each honeypot has an independent processing unit, it may introduce computational overhead. In addition, HoneyCam only provides the option to use 360-degree videos, but sometimes the 360-degree image is good enough, especially for the honeypots that mainly display static scenes. The 360-degree image is easier and cheaper to make compared with 360-degree video since 360-degree video requires the use of special devices.

To sum up, compared with Siphon, *SweetCam* avoids the real device cost by using fake video and also fulfills the requirements for attacker interaction. Compared with HoneyCam, *SweetCam* processes the media at the front-end rather than the processing unit, making it more scalable, and also exempting the cost of setting up the processing unit and avoiding the possibility of single-point failure if the requests become massive.

5 CONCLUSION

In this work, we propose *SweetCam*, an open-source, medium-interaction IP camera honeypot that supports the emulation of RTSP, SSH, and HTTP protocols. The honeypot is extensible and has minimal configuration overhead in comparison with previous IP camera honeypots. We evaluate *SweetCam* by deploying it in different geographical locations for three weeks and receiving a large number of traffic from 5,924 unique IP addresses. We observe diverse attacks like brute-force attacks, web scrapers, malware, and multistage attacks.

As future work, we aim to extend *SweetCam* with a more heterogeneous user interface to provide better feedback for the adversaries and increase the deception. Besides, the deployment part, *SweetCam* requires a manual process on the cloud platform. To simplify this procedure for longitudinal studies, we aim to implement a seamless deployment configuration that simplifies the deployments.

REFERENCES

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*. 1093–1110.

[2] Warren Cabral, Craig Valli, Leslie Sikos, and Samuel Wakeling. 2019. Review and analysis of cowrie artefacts and their potential to be used deceptively. In *2019 International Conference on computational science and computational intelligence (CSCI)*. IEEE, 166–171.

[3] Chongqi Guan, Xianda Chen, Guohong Cao, Sencun Zhu, and Thomas La Porta. 2022. HoneyCam: Scalable High-Interaction Honeypot for IoT Cameras Based on 360-Degree Video. In *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 82–90.

[4] Juan David Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. 2017. Siphon: Towards scalable high-interaction physical honeypots. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. 57–68.

[5] Dan Gunter. 2017. *Threat Hunting With Python Part 2: Detecting Nmap Behavior with Bro HTTP Logs.* https://www.dragos.com/blog/industry-news/threat-hunting-with-python-part-2-detecting-nmap-behavior-with-bro-http-logs/

[6] Muhammad A Hakim, Hidayet Aksu, A Selcuk Uluagac, and Kemal Akkaya. 2018. U-pot: A honeypot framework for upnp-based iot devices. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 1–8.

[7] Jörg Krause and Jörg Krause. 2017. Language Components of Pug. *Programming Web Applications with Node, Express and Pug* (2017), 89–114.

[8] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. 2017. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat* 2017 (2017), 1–11.

[9] Banyatsang Mphago, Ontiretse Bagwasi, B Phofuetsile, and H Hlomani. 2015. Deception in dynamic web application honeypots: Case of glastopf. In *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer ..., 104.

[10] Banyatsang Mphago, Dimane Mpoeleng, and Shedden Masupe. 2017. Deception in web application honeypots: case of Glastopf. *International Journal of Cyber-Security and Digital Forensics* 6, 4 (2017), 179–185.

[11] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. 2019. Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2702–2733.

[12] Lindsey O'Donnell-Welch. 2022. *Linux Botnet Targets Weak SSH Server Credentials.* https://duo.com/decipher/linux-iot-botnet-targets-weak-ssh-server-credentials

[13] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the rise of IoT compromises. *Emu* 9, 1 (2015).

[14] Deutsche Telekom AG Honeypot Project. [n. d.]. T-Pot: A Multi-Honeypot Platform.

[15] The Honeynet Project. 2021. The Honeynet Project.

[16] Rajesh Kumar Shrivastava, Bazila Bashir, and Chittaranjan Hota. 2019. Attack detection and forensics using honeypot in IoT environment. In *Distributed Computing and Internet Technology: 15th International Conference, ICDCIT 2019, Bhubaneswar, India, January 10–13, 2019, Proceedings 15*. Springer, 402–409.

[17] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2021. Open for Hire: Attack Trends and Misconfiguration Pitfalls of IoT Devices. In *Proceedings of the 21st ACM Internet Measurement Conference (IMC '21)*. Association for Computing Machinery, New York, NY, USA, 195–215. https://doi.org/10.1145/3487552.3487833

[18] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2021. RIoTPot: a modular hybrid-interaction IoT/OT honeypot. In *26th European Symposium on Research in Computer Security (ESORICS) 2021*. Springer, Springer, Darmstadt, Germany.

[19] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2022. Deceptive directories and "vulnerable" logs: a honeypot study of the LDAP and log4j attack landscape. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 442–447. https://doi.org/10.1109/EuroSPW55150.2022.00052

[20] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2022. Interaction Matters: A Comprehensive Analysis and a Dataset of Hybrid IoT/OT Honeypots. In *Proceedings of the 38th Annual Computer Security Applications Conference* (Austin, TX, USA) *(ACSAC '22)*. Association for Computing Machinery, New York, NY, USA, 742–755. https://doi.org/10.1145/3564625.3564645

[21] Suramya Tomar. 2006. Converting video formats with FFmpeg. *Linux journal* 2006, 146 (2006), 10.

[22] Emmanouil Vasilomanolakis, Shreyas Srinivasa, Carlos Garcia Cordero, and Max Mühlhäuser. 2016. Multi-stage attack detection and signature generation with ICS honeypots. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Istanbul, Turkey, 1227–1232. https://doi.org/10.1109/NOMS.2016.7502992

[23] Alexander Vetterl and Richard Clayton. 2019. Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*. 1–13. https://doi.org/10.1109/eCrime47957.2019.9037501

[24] Ruchi Vishwakarma and Ankit Kumar Jain. 2019. A honeypot with machine learning based detection framework for defending IoT based botnet DDoS attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 1019–1024.

[25] Binglai Wang, Yu Dou, Yafei Sang, Yongzheng Zhang, and Ji Huang. 2020. IoTCMal: Towards a hybrid IoT honeypot for capturing and analyzing malware. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[26] Armin Ziaie Tabari and Xinming Ou. 2020. A multi-phased multi-faceted iot honeypot ecosystem. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2121–2123.