PARVP: Passively Assessing Risk of Vulnerable Passwords for HTTP Authentication in Networked Cameras

Jay Anand, Arunan Sivanathan, Ayyoob Hamza, Hassan Habibi Gharakheili
School of Electrical Engineering and Telecommunications, UNSW Sydney, Australia
Emails: {j.anand@student., a.sivanathan@, m.ahamedhamza@, h.habibi@}unsw.edu.au

ABSTRACT

Networked cameras continue to be an attractive target of cyber-attacks and therefore present huge risks to organizations. The use of vulnerable credentials (manufacturers default or publicly known) by these devices remains a primary concern for network and cybersecurity teams. This paper aims to assist enterprise network operators to systematically and passively assess the risk of using default credentials or vulnerable authentication schemes for directly accessing connected cameras. Our contributions are two-fold: (1) We analyze HTTP traffic traces of enterprise-grade network cameras (sourced from popular manufacturers including Cisco, Axis, and Pelco), identify the signature of their authentication techniques, including Basic, regular Digest, and Web Service Security (WSS), extracted from request packets, and develop a system with an algorithm (*PARVP*) for automatic and passive assessment of authentication risks; and (2) We apply PARVP to traffic traces of about 1.4 million HTTP authentication sessions selectively collected from network traffic of more than 1000 cameras (in our university campus network) during three weeks, and draw insights into risks, including cameras that accept default passwords (though hashed) and camera controllers that reveal passwords (though obsolete) by insecure authentication.

CCS CONCEPTS

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

1 INTRODUCTION

Stories about compromised networked cameras continue to appear on news [1, 4]. The most prominent one was highlighted in the large-scale distributed denial-of-service (DDoS) attack in 2016, sourced by the Mirai botnet, which shut down parts of the Internet infrastructure. That attack was particularly successful since the users of cameras (and some other victim IoT devices) continued to rely on default usernames and passwords [18], which was the entry point for Mirai. Cameras with default passwords [31] remain an attractive target to both unskilled and experienced cybercriminals. The attackers incentive varies from obtaining credit card information of customers in shops and selling video footages of intimate places to exploiting these devices for DDoS attacks or even covert cryptocurrency mining [23].

Enterprises tend to increasingly invest in physical security solutions with a specific focus on deploying more cameras to create a safer environment for their staff and customers. Manufacturers often ship the cameras with a default username and password (publicly known [2, 5]) so that users and administrators can gain initial access to configure them. Those default passwords remain unchanged despite basic security guidelines, either for convenience (especially in large-scale deployments with thousands of cameras) or forgetfulness. Even with the change of default passwords, networked cameras can still impose cyber risks, because of communicating via unencrypted services like HTTP on the network.

Existing commercial tools (primarily designed for general IT assets like enterprise servers) *actively* scan the network by sending pre-configured test traffic to individual connected devices and identify their risks and vulnerabilities [32]. However, active assessment approaches may not be necessarily desirable for a heterogeneous set of OT[1]/IoT assets (like cameras, printers, sensors) deployed at scale. Because incompatible queries sent by scanners may render them dysfunctional devices or high volumes of scan traffic may create congestion on their specific network segments – customizing scans per asset or segment can be practically challenging. Lastly, vulnerability scanners do not operate continuously (often scheduled quarterly, monthly, or weekly depending on compliance requirements and best practice cybersecurity frameworks [9, 28] being followed), and hence some temporary devices may get missed. Therefore, *passive* solutions sound more appealing to OT environments by silently and continuously analyzing their network traffic.

This paper[2] develops a passive and systematic risk assessment method with a specific focus on HTTP-based authentications in networked cameras. Our **first** contribution (§2) analyzes packet traces of HTTP authentication for popular enterprise-grade cameras sourced from Cisco (ten models), Axis (two models), and Pelco (two models). We highlight their authentication methods and identify specific signatures in authentication queries and responses. We develop a system empowered by "selective" inspection of network packets (enabled by leveraging formal MUD[3] behavioral profile of IoT devices), and an algorithm (PARVP) to automatically and passively determine whether cameras are using vulnerable credentials (due to the use of well-known or manufacturers default passwords). Our **second** contribution (§3) applies PARVP to traffic traces of 1.4 million HTTP authentication sessions from 1108 live cameras collected from our university campus network during three weeks. We draw insights into the risks of this network by detecting cameras that accept default passwords (though hashed) and controllers that reveal passwords (though obsolete) by insecure authentication.

DAI-SNAC'21, December 7, 2021, Virtual Event, Germany

ACM ISBN 978-1-4503-9136-8/21/12…$15.00
https://doi.org/10.1145/3488661.3494031

[1] Operational Technology
[2] Funding for this project was provided by CyAmast Pty Ltd.
[3] Manufacturer Usage Description (RFC8520) specifies the indented behavior of devices by a structured set of network flows.

2 AUTHENTICATION IN CAMERAS & OUR PASSIVE RISK ASSESSMENT

Networked cameras are designed to provide video surveillance, ensuring physical security. Their decreasing cost and reliable performance have led them to be deployed in various environments such as university campuses, hospitals, commercial buildings, and construction sites. To prevent an unauthorized user from remotely accessing the connected cameras, they need to authenticate valid users. Therefore, manufacturers of these devices, depending upon their size and maturity, are expected to empower their cameras with robust and secure authentication methods. In this section, we begin with a brief discussion on ways of remote access and authentication offered by enterprise-grade networked cameras, as shown in the bottom section of Fig. 1. We next present our method for assessing the risk of access authentication in these cameras, as illustrated in the top section of Fig. 1.

Remote access and authentication: Commercial networked cameras often allow their legitimate users to access video footage (live and/or recorded). To grant access, each camera often acts as an HTTP server, authenticating requests from clients. Best security practices, however, suggest that web servers should operate over HTTPS and not HTTP. We note that employing HTTPS for cameras deployed at scale in enterprise networks can be practically challenging. This is primarily because, in HTTPS, the server (cameras) must present its SSL/TLS certificates to connecting clients, verifying its identity and thus preventing hackers from intercepting any exchanged data. Certificates need a trusted authority to sign them or to be self-signed by the corresponding camera (the server) [29]. Obtaining individual certificates from trusted authorities is deemed expensive or at least challenging by network operators due to additional engineering and operational efforts (*e.g.,* creating publicly registered host-name [7]) required for thousands of installed cameras. Self-signed certificates are also not easy to adopt, as they entail operators to store them (and make them trusted) on every controller (client) that intends to access these cameras.

To overcome these challenges, some well-established manufacturers have introduced the concept of cloud-managed networked cameras [21]. A cloud-based software centralizes the control of a network of cameras, allowing operators to interact with their cameras more seamlessly and securely, manage user permissions, perform security checks, and more. This added security introduces its own technical and economic challenges, including: (a) specific configuration required per each camera, (b) provisioning additional Internet bandwidth, (c) securing the communication between cameras and the cloud server, and (c) paying subscription fees (vendor lock-in).

Therefore, most enterprises typically tend to rely on simpler methods like segmentation ("*air-gapped*" network) [22] to manage the cybersecurity risk of their operational devices and often isolate their cameras at Layer 2 (connecting them to specific VLANs) or at Layer 3 (configuring them on certain subnets). That way, they feel comfortable to continue using HTTP for accessing their cameras from within the enterprise network. Air-gapping, however, cannot be considered as a completely secure measure because it can be physically breached by a third-party networked laptop/phone or USB drive [13].

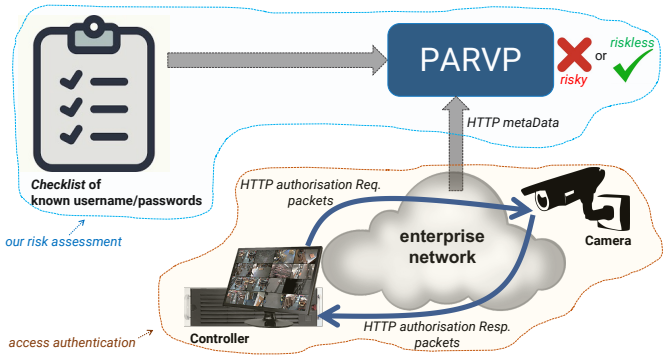


Figure 1: System architecture of our solution for assessing the risk of HTTP authentication.

2.1 HTTP Authentication

There exist several standard schemes [17] of HTTP authentication available for manufacturers to implement for their networked cameras, each differing in their operation and security. Certain schemes such as OAuth are used to facilitate third-party access delegation across web applications without sharing the passwords [14], while others like HOBA[4] require a digital signature (combination of public and private keys) for each client without needing a password database on the server [8]. Before accessing the camera, the client (remote controller) needs to determine the authentication scheme supported by the HTTP server running on the camera.

HTTP authentication schemes involve four steps of exchanging challenge-response between the client and server (Appendix A) – the server specifies the expected authentication scheme. In commercial networked cameras [6], two schemes, namely Basic and Digest [10] are commonly used.

Basic & Digest authentication: While both Basic and Digest methods can authenticate users, the Digest method provides greater security and is therefore recommended for unencrypted channels (when TLS is not available) [10]. In the Basic authentication, a request contains a header field in the form of "`Authorization: Basic <credentials>`", where credentials are the Base64 encoded version of username and password joined by a single colon. The Base64 encoding is easily reversible and thereby insecure over HTTP. Digest authentication (RFC2617), on the other hand, applies several steps to compute a *hashed* response value that is used for authentication – details can be found in Appendix B.

Digest authentication has undergone many revisions since its inception [30]. Today, developers are provided with various options of advanced hashing algorithms like SHA-256 (instead of MD5) to enhance the collision resistance. Also, the classic steps (in Appendix B) can be adjusted to embed more arguments for computing the hashed response. These requirements (and options) are relayed by setting a field called `qop`[5] in the `WWW-Authenticate` header sent (by the server) during initial steps of HTTP authentication.

[4]HTTP Origin Bound Authentication
[5]The "quality of protection" (qop) field helps clients/servers check the integrity of message bodies.

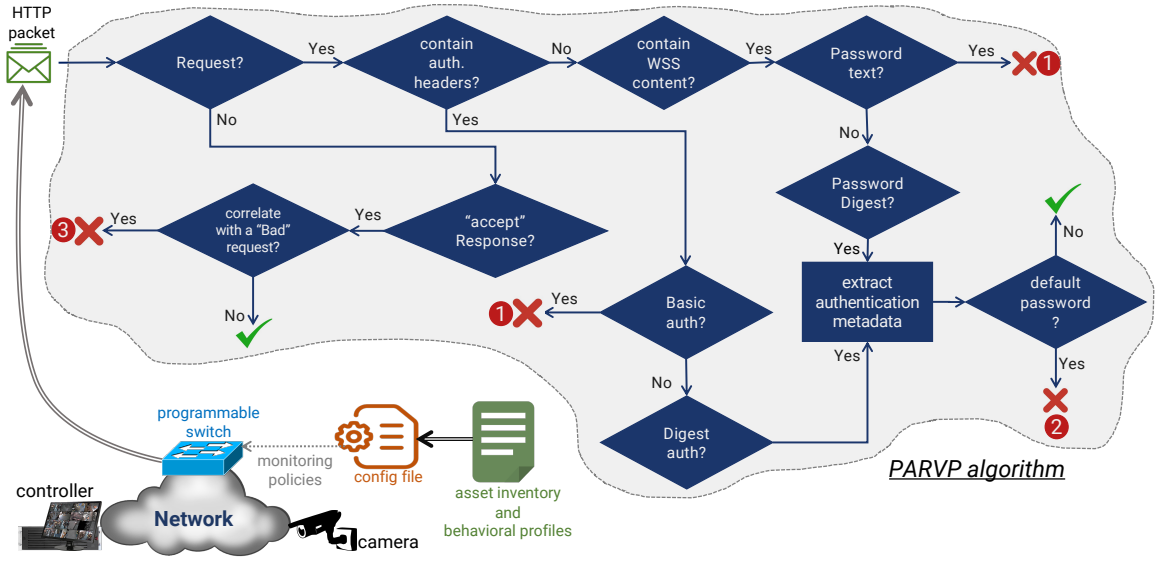


Figure 2: PARVP system & algorithm.

Web Service Security (WSS): WSS is an extension to the SOAP messaging protocol (typically encapsulated in HTTP packets) to apply security features like certificates and/or credentials to Web services. `UsernameToken` is a key security feature in WSS, responsible for carrying credentials like username and password in XML format. It supports both plaintext and digest passwords. Appendix C describes how a hashed response is produced. Developers may choose to use SOAP-based communication over HTTP, as SOAP offers a rich set of libraries and extensions for transferring diverse data. In the context of networked cameras, controls on tilt and zoom can be embedded within an XML tag, rather than using a combination of HTTP headers, forms, and payloads. This flexible approach led manufacturers to agree on a standard, called *ONVIF*[6], to leverage SOAP-based communication for data exchange across multi-vendor platforms.

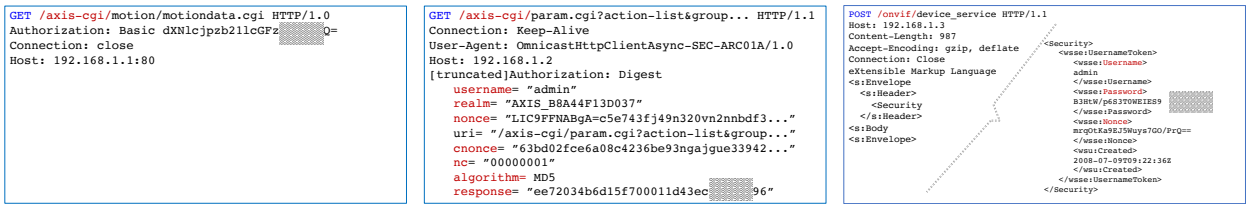
2.2 Risk Assessment Algorithm & System

Fig. 2 illustrates a high-level overview of our system for passively assessing the risk of vulnerable passwords (PARVP). We use a decision tree-based algorithm (*i.e.,* the PARVP algorithm in Fig. 2) to determine how an HTTP authentication packet has to be processed. First, we determine whether the packet is a request or response. If the packet is neither a request nor a response, it will be dropped (not shown in Fig. 2 for brevity). Given a request packet, we extract the authentication scheme used and its corresponding metadata from the HTTP packet header or content. If the authentication scheme is Basic or plaintext WSS, the user credentials can be easily decoded or readily obtained, and hence a risky scheme (red "×(1)" marks). In case of a more secure authentication scheme (*i.e.,* digest), we perform our password check: the authentication metadata of the request packet along with individual passwords in our prepopulated checklist (default and publicly known) are passed through the hashing process of the authentication scheme. Note that metadata elements like `Nonce` can change dynamically for every authentication session, and therefore checklist digest values need be computed per each request packet (cannot be pre-computed). If none of the checklist passwords yield the same outcome (digest), then the request is secure (green "✓" mark), otherwise a risky password is identified (red "×(2)" mark). Lastly, if an authentication response is paired with an accepted risky request (containing an encoded/plaintext password or digest of a default/known password), then the third type of cyber risk is flagged (red "×(3)" mark). Packets that do not contain either Basic or Digest authentication content will be dropped (not shown in Fig. 2 for brevity) by the PARVP algorithm.

Selective Packet Inspection: It is important to note that we only inspect selected HTTP packets from specific cameras on the network. Our method can scale to high throughput (*e.g.,* tens of Gbps) of network traffic since it does not inspect every packet. Selective deep packet inspection is facilitated by employing a programmable switch (*e.g.,* OpenFlow-based or P4-based) that sits parallel to the operational network, receiving a copy of the entire network traffic and mirroring the selected packets for passive inference. Monitoring policies (target devices and their selected packets) can be defined statically via a configuration file or dynamically via API calls, specifying IP address and/or MAC address and/or VLAN id of devices, along with the transport-layer service identity such as protocol and port number (*e.g.,* inbound and/or outbound `TCP/80`) or other fields of packet headers. Note that IoT device manufacturers may choose to operate the HTTP service on a non-standard port number other than `TCP/80`. In this work, the network operator is assumed to have all network assets classified (asset inventory), and their behavioral profile (compatible with the MUD[7] standard) is determined [16] prior to risk assessment so that a config file is proactively created, as shown at the bottom of Fig. 2.

[6]Launched in 2008, ONVIF (the Open Network Video Interface Forum) is an open industry forum with the goal of enabling interoperability between physical security products sourced from different manufacturers.

[7]A formal description of network behavior that can be translated to a set of flow rules enforced at run-time – traffic that conforms to these rules can be allowed, while unexpected traffic is dropped or inspected for potential intrusions [15].



(a) Basic authentication. (b) Regular Digest authentication. (c) WSS Digest authentication (ONVIF).

Figure 3: Signature of authentication packets: (a) Basic, (b) regular Digest, and (c) WSS Digest authentication (ONVIF standard); We obfuscated IPv4 addresses and (encoded/hashed) passwords for privacy reasons.

Table 1: Summary of our dataset.

	Total			Accepted		
	sessions	cameras	controllers	sessions	cameras	controllers
Basic	282,292	1,108	36	0	1,103	19
Digest	1,100,625	1,108	36	388,176	1,103	19

Table 2: Accepted authentication sessions.

	Cisco		Axis		Pelco	
	sessions	cameras	sessions	cameras	sessions	cameras
Basic	0	0	0	0	0	0
Digest	353,994	722	34,182	381	0	0
Digest-risky	0	0	38	6	0	0

Table 3: Rejected authentication sessions.

	Cisco		Axis		Pelco	
	sessions	cameras	sessions	cameras	sessions	cameras
Basic	0	0	282,292	184	0	0
Digest	705,572	695	6,301	30	576	4
Digest-risky	0	0	0	0	0	0

Signatures: In order to correctly extract authentication metadata, we developed signatures in our lab by offline processing of packet captures. Our set of signatures currently comprises three widely-used authentication schemes: HTTP Basic, HTTP Digest, and WSS UsernameToken plaintext/Digest (used by ONVIF-compatible devices) – it can be readily extended to accommodate other authentication schemes emerging in the future. Fig. 3 provides a visual representation (*i.e.,* Wireshark view) of packet signatures for each of these three schemes. The signature for HTTP Basic and Digest schemes are defined by the use of an Authorization header as seen in Fig. 3(a) and Fig. 3(b), respectively. The authorization header describes the scheme in use and provides all corresponding metadata required to authenticate the client. WSS UsernameToken uses an XML structure to encapsulate all authentication metadata inside the "`UsernameToken`" tag of the HTTP content, as shown in Fig. 3(c). Note that the default hash method used in WSSE (and ONVIF) is SHA-1 (Appendix C), but it is not explicitly displayed in the packet signature.

3 EVALUATION RESULTS

We prototyped PARVP on a research testbed receiving bidirectional traffic of production cameras in our university campus. The IT department of our university provisioned a full mirror (both inbound and outbound) of traffic (at an average rate of ≈20 Gbps) for more than 20,000 network-connected devices (IT and IoT/OT) to our testbed from their core router. We obtained appropriate ethics clearances (UNSW Human Research Ethics Advisory Panel approval number HC190171) for this study. This experimental setup aims to demonstrate the efficacy of our proposed method with real traffic and draw insights into the cyber risks associated with the configuration of cameras and their corresponding controllers accessing them over the enterprise network.

Password Checklist: We constructed a checklist of 12,200 unique passwords obtained from four public sources, including: (a) passwords used in the Mirai botnet [24], (b) an extensive list of simple, yet probable passwords identified for security testing [25], and (c) two public lists of default passwords that are known to be in use for different types of networked cameras [5, 31]. These passwords are a combination of default (passwords used by manufacturers to enable installation/setup) and vulnerable (simple English and or numerical strings that are common and can be easily guessed). It is important to note that the checklist is extensible and can be augmented to add new default and vulnerable passwords.

Prototype Implementation: Physical surveillance cameras (14 different models[8] sourced from three manufacturers, namely Cisco, Axis, and Pelco) operate on a dedicated VLAN (consisting of six subnets of size /24) of the campus network. Each camera on average generates a traffic rate of ≈2 Mbps. We had separately generated the MUD profile of each camera type by passively analyzing their network traffic [16], and found that all cameras offer the HTTP service on the standard transport-layer port `TCP/80`. Hence, the config file specifies a filter on the `TCP/80` traffic of the specific VLAN id of the camera segment. We used NoviSwitch 2122 (a fully OpenFlow 1.3 compliant SDN switch) and the Floodlight controller (v1.2) for our system. Floodlight exposes northbound RESTful APIs to receive monitoring policies. Selected packets (mirrored by the SDN switch) are stored in PCAP files. We developed a Python script, utilizing the `Scapy` library to parse PCAP files and arrange matching HTTP requests and responses as JSON objects in a set of JSON files. Each request/response JSON object was then fed into another Python script, which implements the PARVP algorithm. Based on the output of the PARVP algorithm, the selected request/response JSON object was updated to reflect the risk type identified.

Traffic Collection and Dataset: For this work, we collected samples of HTTP authentication request and response packets of 1108 cameras on our testbed during three weeks in May and Jun 2021. Each sampling period, capped at 2 GB worth of data, focused on an /24 subnet within the specific VLAN dedicated for the cameras. In total, we collected traces of selected packets across 20 sampling

[8]The exact model of cameras is not revealed for privacy reasons.

periods, each scheduled (using a `cron` task) to commence at two different times (peak and off-peak hours) of selected weekdays.

Following traffic collection, we created a dataset of JSON records, each containing metadata of a matching pair of HTTP request/response. Our metadata includes the IP address of cameras and their respective controllers, attempted credentials, authentication scheme, HTTP response code, and the request's timestamp. Table 1 summarizes our dataset (total and accepted sessions) across the two main authentication methods (*i.e.,* Basic and regular/WSS Digest). In total, our dataset contains 1,380,000 authentication sessions, sourced from 36 controllers to 1108 cameras. It can be seen that about 80% of all access attempts use the digest method for authentication (hashed password presented), while the remaining 20% use the basic authentication (encoded password presented), which is not recommended. Another observation is that more than 70% of all authentication attempts are rejected. Interestingly, 17 controllers are consistently rejected from access to cameras – they seem to be outdated and continue using obsolete methods (Basic authentication sessions) and/or changed credentials (≈65% of digest authentication sessions). Also, we note that no camera accepts the basic authentication, highlighting baseline "cyber hygiene" practices implemented across this network of cameras. Lastly, five cameras (an Axis and four Pelco) are found to consistently receive bad requests from their respective controllers, suggesting outdated configurations.

3.1 Vulnerable Cameras

Table 2 focuses on accepted fraction of authentication sessions in our dataset, and shows how they are distributed across Basic, Digest, or Digest-risky methods. Note that Digest-risky refers to those Digest sessions wherein the presented password is found in our password list. We can see that six Axis cameras are identified at Digest-risky (cells highlighted in red), meaning that they indeed accept default passwords for authentication (*i.e.,* risk types "×(2)" and "×(3)" in Fig. 2). All of these risky cameras share the same credentials `root`/"`a-default-password`" [9]. This finding highlights how enterprises sometimes may overlook to enforce baseline security best practices like changing default passwords on devices prior to deploying them on the network.

Table 3 focuses on the rejected portion of sessions in our dataset. We can see that all Basic authentication attempts (*i.e.,* risk type "×(1)" in Fig. 2) were made towards 184 Axis cameras, suggesting a set of probably outdated controllers which initiate invalid requests. Furthermore, we observe that 67% of Cisco cameras, 15% of Axis cameras, and all Pelco cameras are target of these rejected sessions over Digest authentication. We found that Axis and Pelco cameras reject an authentication session wherein the controller fails to format the request packet correctly or supplies an invalid credential, while Cisco cameras typically fail to authenticate due to limited resources available on the device (will be discussed in §3.2). Lastly, no Digest-risky sessions (those that offer default/known passwords) are found among the rejected subset of our records.

Extended Assessment with Augmented List: Further analysis of the rejected sessions over Basic authentication (the first row in Table 3) revealed that all of them share a credential `admin`/

[9]We obfuscated this default password for privacy reasons.

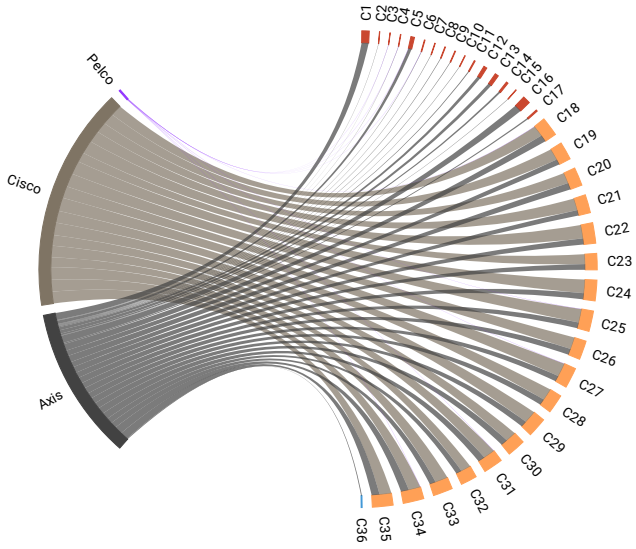


Figure 4: Thirty-six controllers (color-coded) attempt to access 1108 cameras: C1-C17 (red) are constantly rejected by cameras; C18-C35 (orange) are accepted sometimes and rejected some other times; and, C36 (blue) is always accepted.

"`some-password`" [10] – this specific password (obfuscated) was not in our original password checklist. The wide usage of this specific credential, found in more than 280K rejected sessions attempting to access 184 Axis cameras, suggests a configuration that is possibly common across a department or even the entire organization. This means that the same credentials might have been used in Digest sessions too, but requires further checks. We, therefore, augment our original checklist by incorporating new passwords (in this case "`some-password`") discovered during the first round of risk assessment process, and perform a second verification with the updated credentials.

Interestingly, we verified our hypothesis – "`some-password`" was used as password for Digest-based authentication with Axis and Pelco cameras (*i.e.,* risk type "×(2)" in Fig. 2). Fortunately, all of these attempts were rejected, highlighting that risky password was obsolete. All of the rejected sessions with Pelco cameras and 86% of the rejected sessions with Axis cameras, offered the pair "`admin/some-password`" for authentication. This again highlights another risk when some organizational passwords (that are unknown publicly) can get leaked through the use of Basic authentication. We reported all of these vulnerabilities (known passwords and weak authentication schemes) to the cybersecurity department of our university (for remedial actions) prior to the submission of this paper.

3.2 Vulnerable Controllers

We now focus on two potential risks (*i.e.,* cyber and operational) associated with controllers accessing or attempting to access the cameras. Cyber risk is identified when an identifiable (plaintext or hashed version of known/default) password is presented for authentication. Operational risk is identified when authentication attempts consistently fail due to incorrect formatting the request

[10]We obfuscated this password for privacy reasons.

or presenting invalid credentials. We categorize (and color-code) the controllers into three groups, including (i) red ones (C1-C17), which are consistently rejected by cameras, (ii) orange ones (C18-C35), which are accepted sometimes and rejected some other times, and (iii) the blue one (C36), which is always successful in HTTP authentication. Fig. 4 visualizes HTTP communications between 36 controllers (C1-C36) and 1108 cameras (Axis, Cisco, and Pelco) in our dataset.

The red controllers only contact Axis cameras and solely offer the obsolete password (*i.e.,* `some-password`) via Basic and/or Digest authentication. In order to mitigate the operational and cyber risks that the red controllers present to this camera network, they probably need to be decommissioned or at least reconfigured/upgraded. The orange controllers tend communicate with all three types of cameras (Axis, Cisco, and Pelco) and display some operational failures (rejected authentication requests). Two controllers (C28, C29) were found transmitting ONVIF-formatted requests to Axis cameras. However, Axis cameras on this network are not yet ONVIF-enabled, hence respond with a `400 Bad Request` status code. Another two controllers (C23, C24) were found sending incomplete HTTP digest authentication metadata (Fig. 3(b)) expected by Axis cameras. In their request packets, they missed to embed `cnonce`, `nc`, `algorithm` and `response` into their requests, hence resulted in HTTP responses with status code 400. Also, we found the orange controllers (C18-C35), communicating with Cisco cameras (ONVIF-enabled), infrequently receive an HTTP response with status code 400 and a payload containing "`max pull point exceeded`" error. Note that each ONVIF request for certain tasks, like tilting the camera lens or accessing the camera feed, creates a pull point resource on the camera which has a limited capacity available to handle concurrent connections. Finally we found that 15 orange controllers occasionally received "`500 Internal Server Error`" responses from their target cameras. Overall, the orange controllers display behaviors indicating some operational risk but not significant cyber risks. Lastly, the only blue controller seems to operate in a riskless manner, with all of its authentication requests successfully responded to by six Axis cameras. However, this controller and its target cameras are not necessarily secured since the authentication requests contain a default password (though it is hashed). Therefore, the configuration of this controller and those six cameras needs to change, mitigating a critical cyber risk.

Further, we investigated to identify the application used by individual controllers for accessing the cameras. We found that 30 controllers provided `User-Agent` in the header of their HTTP requests made to the cameras, while the other six did not reveal their agent. Three agents including `gSOAP` (from 10 controllers), `omnicast` (from 19 controllers), and `Chrome` (from a controller) are identified. We note that agents like `gSOAP` and `omnicast` are typically used in management applications of IP (Internet Protocol) cameras [11, 12]. Interestingly, the controller C36 accesses the Axis cameras from the `Chrome` web browser, not from the vendor-supplied management software.

4 RELATED WORK

Risk Assessment for IoT Devices: Prior efforts to assess the risk of IoT devices are relatively broad and diverse in their scope and techniques. Work in [20] tests a suite of active and passive risk evaluation techniques on consumer devices to quantify the vulnerabilities specific to key security pillars (confidentiality, integrity, access control) as well as the ability to reflect attacks. The authors propose a subjective color-coded scheme that highlights the security rating of individual devices tested. An extensive measurement study [27] focuses on the privacy risks of consumer IoTs. The authors analyze the network traffic of IoT devices to determine any information exposures like exchanging unencrypted content with the Internet or inferred behavior of devices. Our paper, instead, focuses narrowly on the authentication vulnerability of enterprise-grade cameras in production networks. We develop a passive risk assessment system to determine whether cameras accept well-known or manufacturers' default passwords via HTTP-based authentication.

Cybersecurity of Connected Cameras: Several research studies [3, 19, 33] on the security posture of connected cameras discuss how they can be threatened, highlighting the compromised publicly available cameras as well as commercial models at most risk.

Authors of [3] developed an analysis framework for actively assessing the risk of networked cameras. They perform a variety of active tests to determine whether a given camera is vulnerable to unauthorized video stream retrieval, device tampering, unauthorized account hijacking or unauthorized capture of data. Out of the five camera models they tested, three were identified to be using default passwords, and four enforced no strict password complexity policy, encouraging vulnerable passwords. Authors of [33] investigated cameras that are completely exposed to the Internet with no password protection. Their work analyzed the information available on the "`insecam.org`" website, which provides live video footage of cameras across the globe. The authors found that more than 20,000 live feeds are available every day, with an average of more than 200 new cameras added daily. Work in [19] analyzed a broad range of vulnerabilities in networked cameras. The authors formulated a diverse set of attack scenarios like adversarial machine learning and brute force. By looking up popular manufacturers on "`shodan.io`" and "`censys.io`" websites, they found more than a million cameras exposed to the Internet – 90% of those cameras communicate over HTTP (not HTTPS).

This paper builds upon prior work. We highlight how unencrypted protocols like HTTP can risk the security of a network of cameras. Our risk assessment findings can help enterprise network operators mitigate the risk of threat actors exploiting default credentials to access a camera.

5 CONCLUSION

Networked cameras are rapidly deployed at scale in enterprises, mitigating physical security risks but introducing cyber risks. This paper developed a system (PARVP) that employs the formal model of networked cameras along with an algorithm that automatically and passively determines whether cameras authenticate securely or not. We tested PARVP against 1.4 million HTTP authentication sessions captured from more than 1000 cameras in a production network. We identified a few risky cameras that accept default passwords and some risky controllers that leak non-default passwords via insecure authentication. Our PARVP can be employed for assessing the authentication risks of any OT/IoT device type and is readily extensible to support other forms of unencrypted authentication emerging in the future.

REFERENCES

[1] ABC News. 2020. Australian security cameras hacked, streamed on a Russian-based website. https://ab.co/3ealY0j.
[2] Ethan Ace. 2021. IP Cameras Default Passwords Directory. https://ipvm.com/reports/ip-cameras-default-passwords-directory
[3] Rana Alharbi and David Aspinall. 2018. An IoT Analysis Framework: An Investigation of IoT Smart Cameras' Vulnerabilities. *Living in the Internet of Things: Cybersecurity of the IoT* (Mar 2018).
[4] Bloomberg. 2021. Hackers Breach Thousands of Security Cameras, Exposing Tesla, Jails, Hospitals. https://bloom.bg/2TalGiG.
[5] A1 SEcurity Camras. 2021. Default Usernames, Passwords and IP Addresses for Security Cameras. https://bit.ly/2VrH97j
[6] Canon. 2018. Useful Tips for Reducing the Risk of Unauthorized Access for Network Cameras. https://bit.ly/3yOgIHk
[7] Digicert. 2015. Replace Your Certificates for Internal Names. https://www.digicert.com/blog/replace-your-internal-name-certificates
[8] Stephen Farrell, Paul E. Hoffman, and Michael Thomas. 2015. HTTP Origin-Bound Authentication (HOBA). RFC 7486. https://doi.org/10.17487/RFC7486
[9] NISt: Joint Task Force. 2020. Security and Privacy Controls for Information Systems and Organizations . https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf
[10] John Franks, Phillip Hallam-Baker, Lawrence C. Stewart, Jeffery L. Hostetler, Scott Lawrence, Paul J. Leach, and Ari Luotonen. 1999. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617. https://doi.org/10.17487/RFC2617
[11] Genetec. 2021. Omnicast VMS . https://bit.ly/3hSLF6w
[12] Genivia. 2021. ONVIF examples . https://bit.ly/3xBmegi
[13] Great, M. Lechtik, and G. Dedola. 2020. Cycldek: Bridging the (air) gap. https://securelist.com/cycldek-bridging-the-air-gap/97157/
[14] Eran Hammer-Lahav. 2010. The OAuth 1.0 Protocol. RFC 5849. https://doi.org/10.17487/RFC5849
[15] Ayyoob Hamza, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2018. Combining MUD Policies with SDN for IoT Intrusion Detection. In *Proc. ACM IoT S&P*. Budapest, Hungary.
[16] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Theophilus A. Benson, Matthew Roughan, and Vijay Sivaraman. 2020. Verifying and Monitoring IoTs Network Behavior using MUD Profiles. *IEEE Transactions on Dependable and Secure Computing* (May 2020), 1.
[17] IANA. 2017. HTTP Authentication Schemes. https://bit.ly/3e8AINa
[18] IoT World Today. 2019. Cybersecurity Lessons Related to IP Security Cameras. https://bit.ly/3rddefa.
[19] Naor Kalbo, Yisroel Mirsky, Asaf Shabtai, and Yuval Elovici. 2020. The Security of IP-Based Video Surveillance Systems. *Sensors* 20, 17 (Aug 2020).
[20] Franco Loi, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford, and Vijay Sivaraman. 2017. Systematically Evaluating Security and Privacy for Consumer IoT Devices. In *Proc. ACM IoT S&P*. Dallas, Texas, USA.
[21] Cisco Meraki. 2020. Architecture and Best Practices . https://bit.ly/3xBmgoq
[22] Neerja Mhaskar, Mohammed Alabbad, and Ridha Khedri. 2021. A Formal Approach to Network Segmentation. *Computers & Security* 103 (Apr 2021), 102162.
[23] Trend Micro. 2018. Exposed Video Streams: How Hackers Abuse Surveillance Cameras. https://bit.ly/3yRq9Gg
[24] Daniel Miessler. 2017. mirai-botnet. https://bit.ly/2VARz4O.
[25] Daniel Miessler. 2018. probable-v2-top12000. https://bit.ly/3r3fuW1.
[26] oasis. 2004. UsernameToken Profile . https://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
[27] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proc. ACM IMC*. Amsterdam, Netherlands.
[28] R. Ross, P. Viscuso, G. Guissanie, K. Dempsey, and M. Riddle. 2021. Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-171r1.pdf
[29] Matthew Rossi. 2020. How to secure a security camera for use over the web . https://www.cctvcameraworld.com/secure-security-camera-for-use-on-web/
[30] Rifaat Shekh-Yusef et al. 2015. HTTP Digest Access Authentication. RFC 7617. https://doi.org/10.17487/RFC7617
[31] Anthony Spadafora. 2021. Default passwords make IP cameras surprisingly easy to hack. https://bit.ly/2UCtFFu
[32] Lance Whitney. 2020. How to find and fix vulnerable default credentials on your network. https://tek.io/3k7wZmB
[33] Haitao Xu, Fengyuan Xu, and Bo Chen. 2018. Internet Protocol Cameras with No Password Protection: An Empirical Investigation. In *Passive and Active Measurement (PAM)*. Berlin, Germany.

Appendix A FLOW OF EVENTS IN HTTP AUTHENTICATION

(1) the client initiates a request to log into the server;
(2) the server will respond with an `HTTP 401 Unauthorized` error along with a `WWW-Authenticate` header explaining how to authenticate correctly. Typically, the server will specify the type of HTTP authentication it expects along with some other required information depending on the scheme;
(3) the client will then attempt to authenticate correctly, formatting the authorization header as required;
(4) if credentials in the authorization header are correct, the server will successfully authenticate the client.

Appendix B HASHED RESPONSE IN DIGEST AUTHENTICATION

(1) username, password, and realm (a string that describes the camera) are passed through a hash function (*e.g.,* MD5) to produce the first hashed value:
`HA1 = MD5(username:realm:password)`;
(2) the HTTP request method (*e.g.,* `GET` or `POST`) and digestURI (endpoint to which the client attempts to connect) of the request are hashed together to produce the second hash value:
`HA2 = MD5(method:digestURI)`;
(3) the two computed hash values (*i.e.,* HA1, HA2), nonce (server-generated random value used to prevent replay attacks), nc (nonce count), cnonce (client generated string value used to prevent plaintext attacks) and qop (quality of protection) are hashed together to produce a final response:
`Response = MD5(HA1:nonce:nc:cnonce:qop:HA2)`.

This response value is presented by the client to the server for authentication.

Appendix C HASHED RESPONSE IN WSS AUTHENTICATION

(1) the client creates a nonce, and encapsulates that within an SOAP XML UsernameToken tag along with other variables such as the date, digest of password, and username. This XML payload is encapsulated within the HTTP request packet sent by the client to the server;
(2) If details are correct, then the server will authenticate the client. The key difference here is that the server does not have to specify a nonce, shifting this task to the client. The server only records these requests, thus mitigating replay attacks. Furthermore, the authentication only requires a pair of request/response, instead of two pairs (discussed earlier under standard HTTP schemes). Password digest is computed by: `Digest = B64Encode(SHA1(B64Decode(Nonce) + Date + Password))`.
Note: According to the WSS UsernameToken profile [26], the password digest is computed by: `Digest = B64Encode(SHA1(Nonce+Date+Password))`. Though it is not explicitly stated a B64Encoded version of the Nonce is sent over the wire. To authenticate users, the Nonce retrieved from the SOAP message must be B64Decoded to get the original binary data use to compute the password digest.