GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6

Bezpečnostní kamera

Maturitní práce

Autor: Vítek Peterka

Třída: 4.A

Školní rok: 2022/2023

Předmět: Informatika

Vedoucí práce: Bc. Emil Miler

Praha, 2023

GYMNASIUM JANA KEPLERA
Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

Student:	**Vítek Peterka**
Třída:	**4. A**
Školní rok:	**2022/2023**
Vedoucí práce:	**Emil Miler**
Název práce:	**Bezpečnostní kamera**

Pokyny pro vypracování:

Cílem práce je vytvořit software pro bezpečnostní kameru založenou na jednodeskovém počítači a sestrojit samotný hardware kamery, na kterém bude software běžet. Kamera bude konfigurovatelná, s možností nastavení zálohování videozáznamu na vzdálený server s end-to-end šifrováním. Zároveň bude možné se na server připojit přímo za účelem sledování živého přenosu, který bude taktéž end-to-end šifrovaný.

Doporučená literatura:

[1] CLEMENTS, Alan. ***Principles of Computer Hardware*****. Oxford University Press, 2006. ISBN: 978-0-199-27313-3**

[2] C.P.A Inc. ***The Computer Programming Bible*****. 2020. ISBN: 978-1-661-84628-2**

URL repozitáře:

https://github.com/kokolem/picctv

student *vedoucí práce*

V Praze dne 29. 9. 2022

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 24. března 2023 Vítek Peterka

Poděkování

Děkuji především svému vedoucímu Bc. Emilu Milerovi za podporu a cenné rady při zpracování této práce, především pak za jeho trpělivost a ochotu.

Dále musím poděkovat Vlastimilovi Čejpovi za jeho technickou asistenci a za rady se sázením samotné práce.

Abstrakt

Maturitní práce pojednává o problematice bezpečnostních kamer, nahrávacích zařízení a jejich zabezpečení. Popisuje úskalí běžně dostupných komerčních řešení a navrhuje alternativu v podobě DIY kamery založené na jednodeskovém počítači a vlastním softwaru. Ten umožňuje end-to-end šifrované zálohování videozáznamu na vzdálený server, jakož i živý přenos.

Klíčová slova

bezpečnostní kamera, end-to-end šifrování, nahrávací zařízení, jednodeskový počítač

Abstract

The thesis deals with the issue of security cameras, video recorders and their security. It describes the pitfalls of commonly available commercial solutions and proposes an alternative in the form of a DIY camera based on a single-board computer and custom software. The latter allows end-to-end encrypted backup of video footage on a remote server as well as live streaming.

Keywords

security camera, end-to-end encryption, video recorder, single-board computer

Obsah

1. Teoretická část

Bezpečnostní kamery a nahrávací zařízení jsou pro svou povahu častým cílem internetových útočníků. Navzdory tomu jsou běžně komerčně dostupná řešení v odborné komunitě notoricky známá svým špatným zabezpečením. Kromě zpřístupnění samotného obrazu, který kamera vidí, může kompromitované zařízení posloužit útočníkovi i jako vstupní vektor pro další průnik do počítačové sítě. Výrobci často bezpečnostní rizika přehlíží a raději se soustředí na vývoj chytrých funkcí, jako jsou detekce osob a rozpoznání obličejů.

1.1 Výrobci bezpečnostních kamer

Podle některých odhadů může svět sledovat až jedna miliarda bezpečnostních kamer. [2] Největšími světovými výrobci jsou pak společnosti Hikvision a Dahua, přičemž dohromady ovládají asi 60 % světového trhu s kamerami. [10] Ve Spojených státech amerických byl oběma firmám z důvodu potenciálních bezpečnostních rizik zakázán prodej nových produktů – obě společnosti jsou státní, vlastněny zcela, respektive z části, Komunistickou stranou Číny. [8] Mezi další významné výrobce patří firmy Bosch, Axis a Uniview. [9]

1.2 Nejznámější bezpečnostní problémy

Čínští výrobci jsou nechvalně proslulí bezpečnostními problémy svých zařízení. Společným jmenovatelem chyb bývá zanedbání běžných bezpečnostních praktik, jako je hashování hesel nebo sanitace vstupu od uživatele.

1.2.1 ICSA-17-124-01

Nejznámější bezpečnostní chybou v Hikvision produktech je takzvaný backdoor. Jedná se o chybu vedenou pod označením ICSA-17-124-01. Po zadání speciální hodnoty parametru do URL adresy bylo možné v některých verzích firmwaru zcela obejít autorizační mechanismus a rovnou přistupovat k citlivým datům, jako je například konfigurační soubor obsahující heslo do uživatelského rozhraní. [1] Ačkoliv výrobce uvádí, že se jednalo o pozůstatek z testovací fáze vývoje, toto vysvětlení není experty považováno za pravděpodobné a jedná se tak spíše o záměrně umístěná zadní vrátka. [4]

1.2.2 CVE-2021-36260

Další ze známých chyb v kamerách a nahrávacích zařízeních Hikvision je CVE-2021-36260. Nedostatečná validace vstupu vede v některých verzích firmwaru ke spuštění kódu přímo na daném zařízení. Tato chyba může postihovat až více než 100 milionů produktů. [5] Jedná se o zvlášť závažný problém, neboť útočník může díky této chybě zcela převzít kontrolu nad chováním systému. [3]

1.2.3 CVE-2021-33044

Bezpečností problémy se nevyhnuly ani firmě Dahua. Chyba vedená jako CVE-2021-33044 způsobuje, podobně jako ICSA-17-124-01, možnost kompletně obejít mechanismus přihlašování uživatelů a přistupovat ke kameře s plným oprávněním. Je tedy možné bez znalosti platných přihlašovacích údajů sledovat záznam kamery a měnit její nastavení. [7]

2. Implementace

2.1 Bezpečnostní architektura

Při navrhování bezpečnostní architektury svého systému jsem se snažil co nejvíce omezit potenciální oblasti útoku. Kamera proto nemá žádné uživatelské rozhraní, do kterého by bylo možné se přihlásit, ani žádné složité funkce chytrého rozpoznání objektů. Naopak jsem se zaměřil na to, co by měla bezpečnostní kamera opravdu umět: Nahrávat video, ukládat ho a mít možnost živého sledování. To vše end-to-end šifrovaně.

Kamera posílá jednotlivé snímky na server zašifrované a podepsané privátním klíčem. V této podobě je server i ukládá, není tedy třeba žádná speciální důvěra v poskytovatele serveru. Kamera zároveň nemá žádné otevřené porty (pokud je neotevře uživatel, například pro vzádelný přistup přes SSH), čímž jsou možnosti pro potenciální útok prakticky nulové.

K serveru se lze připojit z prohlížečové aplikace pro živý přenos. Ta není hostovaná přímo na serveru, neboť by útočník mohl nahradit její kód svým a exfiltrovat šifrovací klíče při dalším použití. Místo toho je třeba mít její lokální kopii. Po nahrání souboru s klíči a zadání hesla a adresy serveru začne probíhat přenos snímků ze serveru do prohlížeče. Server je posílá přesně tak, jak mu přicházejí z kamery, tedy zašifrované. Přímo v prohlížeči proběhne rozšifrování a následné zobrazení.

Mezi serverem a klienty probíhá autentifikace heslem. V případě kamery je heslo náhodně vygenerováno, prohlížečová aplikace ho získá jako vstup od uživatele. Aby hesla nemohl útočník odposlouchávající síťovou komunikaci zjistit, je nutné použít TLS, ať už s certifikátem od veřejné certifikační autority, nebo self-signed (v takovém případě je ale potřeba přesvědčit prohlížeč a operační systém kamery, aby certifikátu věřili).

2.2 Model důvěry

Nejzranitelnějším článkem v řetězci je nahrávací server, neboť má otevřený port a útočník se ho tak může pokusit zneužít. Proto je mu ostatními částmi dávána prakticky nulová důvěra. Kompromitovaný server může přestat zpracovávat záznam z kamery, ale to je vše. Díky šifrování a kryptografickému podepisování jednotlivých snímků nemůže útočník záznam číst, ani ho (aniž by to druhá strana poznala) měnit.

Ačkoliv je to nepravděpodobné, teoreticky by mohlo dojít i ke kompromitaci samotné kamery. V takovém případě by útočník mohl změnit, co kamera posílá, a taková data podepsat správným klíčem. Stále by ale nebylo možné číst už nahrané záznamy.

K tomu by bylo potřeba, aby útočník získal soubor s klíči pro dešifrování a prolomil uživatelovo heslo.

2.3 Použitá kryptografie

Pro implementaci kryptografických funkcí jsem využil knihovnu `libsodium`.

2.3.1 Šifrování videozáznamu

Šifrování snímků z kamery je asymetrické, tedy existuje rozdílná sada klíčů pro šifrování (tyto klíče zná kamera) a dešifrování (tyto klíče jsou drženy v tajnosti) záznamu.

Pro implementaci jsem použil modul `crypto_box`. Ten umožňuje tzv. autentizované šifrování, díky čemuž nemůže útočník šifrovanou zprávu nejen přečíst, ale ani změnit.

Kamera zná svůj privátní klíč a veřejný klíč budoucího příjemce. Na základě toho vypočte třetí klíč. Příjemce zná veřejný klíč kamery a svůj privátní klíč. Na základě toho opět vypočte třetí klíč, stejný, jaký dostala svým výpočtem kamera. Tento sdílený klíč je možné použít k ověření toho, že s daty nebylo manipulováno. [6]

Klíče pro dešifrování záznamu jsou dále chráněny uživatelovým heslem – z hesla je hashovací funkcí Argon2i odvozen klíč, kterým jsou zašifrovány. Pro jejich použití k dešifrování záznamu je tedy třeba znát heslo. Jedná se o princip zvaný key wrapping. Pro implementaci jsem použil modul `secretbox`.

2.3.2 Autentifikace

Server nezná hesla jako taková, ale pouze jejich hash. Pro ověření vypočte hash hesla, které mu poslal klient a ten porovná s hashem, který má uložený. Pro implementaci jsem použil hash Argon2id pomocí modulu `pwhash`.

2.4 Hardware kamery

Jako základní stavební jednotku jsem pro jeho dostupnost a všestrannost vybral jednodeskový počítač Raspberry Pi 4 Model B. Má dostatečný výkon pro šifrování a velký výběr kamerových modulů, které k němu lze připojit. Ačkoliv lze použít libovolný jiný, jako kamerový modul jsem nakonec zvolil Arducam 5MP OV5647 IR-CUT Pi Camera. Nad oficiálními moduly vyniká schopností nočního i denního režimu a integrovanými infračervenými světly.

2.5 Použité programovací jazyky

I při výběru programovacího jazyka jsem myslel na bezpečnost. Proto jsem se vyhnul jazykům s nezabezpečeným přístupem k paměti jako C a použil raději Python. Je pro něj dostupná knihovna `libsodium` (přes wrapper `pynacl`) i oficiální Raspberry Pi knihovna pro práci s kamerou `picamera2`. Zároveň je kód lehce čitelný a ověřitelný, což je z hlediska bezpečnosti také důležité.

Pro prohlížečovou aplikaci bylo nutné použít JavaScript (pro který má knihovna `libsodium` také podporu).

2.6 Přenos dat

Jako protokol pro přenos dat mezi jednotlivými částmi projektu jsem vybral WebSocket. Jde o protokol pro přenos dat s minimálním zpožděním založený na TCP. Prohlížeče ho podporují nativně, v Pythonu jsem použil knihovnu `websockets`.

3. Technická dokumentace

Projekt je rozdělen na čtyři části: kamera (`camera`), server (`server`), obslužné nástroje (`utils`) a prohlížečová aplikace pro živý přenos (`viewer-live`).

3.1 Vygenerování a úprava konfiguračních souborů

Kód pro generování konfiguračních souborů se nachází v adresáři `utils`. Následující ukázky předpokládají, že se v tomto adresáři nacházíte i vy.

Nejprve nainstalujte závislosti: `pip install -r requirements.txt`.

Konfigurační soubory lze vygenerovat za použí scriptu, takto: `python generate_configs.py`.

Script se po spuštění zeptá na heslo a vytvoří tři soubory: `server_config.json`, `camera_config.json` a `viewer_config.json`.

3.1.1 Nastavení serveru

V souboru `server_config.json` dle potřeby upravte tyto hodnoty:

- `max_recording_file_size`: Maximální velikost jednoho souboru se záznamem z kamery v bytech
- `max_recording_files`: Maximální počet souborů se záznamem, které budou najednou uloženy na disku
- `records_directory`: Adresář, do kterého se budou průběžně ukládat soubory se záznamem
- `server_port`: Port, na kterém bude server poslouchat

Ve výchozím nastavení bude server poslouchat na portu 8765 a záznam z kamery ukládat do adresáře `records`. Jeden soubor bude mít maximálně 20 Mb a zároveň bude uloženo maximálně 5 souborů. Velikost adresáře records tedy nikdy nepřesáhne $5 * 20$ Mb $= 100$ Mb.

3.1.2 Nastavení kamery

V souboru `camera_config.json` dle potřeby upravte tyto hodnoty:

- `video_height`: Výška přenášeného obrazu v pixelech
- `video_width`: Šířka přenášeného obrazu v pixelech. Kombinace výšky a šířky musí být podporována hardwarem připojeného kamerového modulu
- `video_quality`: Číslo od 0 do 4, čím vyšší, tím nižší úroveň komprese
- `server_address`: Adresa serveru. Pokud budete používat TLS, musí začínat `wss://`

Ve výchozím nastavení bude kamera posílat obraz nejvyšší kvality o rozlišení 972 x 1296 na server na adrese `ws://192.168.0.2:8765`.

3.2 Spuštění serveru

Kód serveru se nachází v adresáři `server`. Následující ukázky předpokládají, že se v tomto adresáři nacházíte i vy.

Nejprve nainstalujte závislosti: `pip install -r requirements.txt`.

Přesuňte do adresáře soubor `server_config.json`.

Spusťte server příkazem `python receiver_server.py`.

3.2.1 TLS

Pokud chcete použít TLS (což je v produkčním prostředí důrazně doporučovanáno), zadejte jako první argument cestu k certifikátu a jako druhý cestu ke klíči. Například takto:
`python receiver_server.py cert.pem key.pem`

3.3 Spuštění kamery

Kód pro kameru se nachází v adresáři `camera`. Následující ukázky předpokládají, že se v tomto adresáři nacházíte i vy. Zároveň předpokládají, že kód spouštíte na Raspberry Pi s připojeným kamerovým modulem. Kód byl testován Raspberry Pi OS Lite (32-bit).

Nejprve nainstalujte pip: `sudo apt install python3-pip`
A následně závislosti: `pip install -r requirements.txt`

Přesuňte do adresáře soubor `camera_config.json` (například pomocí programu Magic Wormhole nebo scp).

Spusťte kameru příkazem `python camera_client.py`.

3.4 Sledování živého přenosu

Kód prohlížečové aplikace pro živé sledování se nachází v adresáři `viewer-live`.
Následující ukázky předpokládají, že se v tomto adresáři nacházíte i vy.

Nejprve nainstalujte závislosti: `npm i`

Aplikaci lze spustit příkazem `npm run dev`. Po spuštění bude vyžadovat zadání tří hodnot:

- `viewer_config.json`: Soubor s klíči k dešifrování obrazu z kamery, byl vygenerován spolu s ostatními konfiguračními soubory scriptem
- Password: Heslo zadané při generování konfiguračních souborů. Bude použito k rozšifrování klíčů a k autentifikaci na server

- Server address: Adresa serveru včetně protokolu (`wss://` pro TLS, `ws://` bez TLS) a portu, tedy například `ws://192.168.0.2:8765`

Po zadání hodnot a kliknutí na tlačítko Start stream začne živý přenos.

3.5 Dešifrování a přehrání uložených nahrávek

Kód pro dešifrování nahrávek se nachází v adresáři `utils`. Následující ukázky předpokládají, že se v tomto adresáři nacházíte i vy a máte už nainstalované závislosti.

Dešifrování lze provést za pomocí scriptu `decryptor.py`. Script bere dva argumenty: soubor s klíči k dešifrování (`viewer_config.json`) a cestu k zašifrované nahrávce. Po spuštění se zeptá na heslo k souboru s klíči. Dešifrovanou nahrávku uloží bez přípony `.enc`.

Nahrávku `nahravka.h264.enc` lze tedy dešifrovat takto:
`python decryptor.py viewer_config.json nahravka.h264.enc`
Po zadání hesla bude nahrávka uložena do souboru `nahravka.h264`.

Dešifrovaná nahrávka je stream H.264 framů. Lze ji přehrát například pomocí programu ffplay: `ffplay nahravka.h264`.

Pro přehrání v běžných přehrávačích (VLC, apod.) je třeba stream zabalit do nějakého kontejneru. To lze udělat například pomocí programu ffmpeg:
`ffmpeg -i nahravka.h264 -c copy nahravka.mp4`

Před zabalením do kontejneru je také možné několik nahrávek spojit do jedné: `cat nahravka1.h264 nahravka2.h264 nahravka3.h264 > spojena.h264`

Závěr

Podařilo se mi identifikovat problémy v komerčních řešeních zabezpečovacích systémů a vyvinout vlastní řešení, které se těmto problémům vyhýbá. Naprogramoval jsem funkční end-to-end šifrovaný systém pro bezpečnostní kameru a vybral a sestavil vhodný hardware pro jeho použití.

V porovnání s komerčními kamerami nelze mou kameru používat ve venkovních podmínkách, protože tomu není uzpůsoben její kryt. Zároveň zabírají zálohy jejího videozáznamu více místa, což je dáno nemožností účinnější komprese na straně serveru, neboť server nemá k nahrávkám z principu přístup.

Projekt by bylo možné rozšířit o podporu více současně připojených kamer a o grafické rozhraní pro generování konfiguračních souborů.

V průběhu práce jsem nabyl hlubších znalostí o šifrovacích algoritmech a o mechanismech komprese a přenosu videa. Celkově práci považuji za velmi povedenou.

Seznam použité literatury

[23] *Hikvision cameras: CISA*. Břez. 2023. URL: https://www.cisa.gov/news-events/ics-advisories/icsa-17-124-01.

[Cos19] Elly Cosgrove. *One billion surveillance cameras will be watching around the world in 2021, a new study says*. Pros. 2019. URL: https://www.cnbc.com/2019/12/06/one-billion-surveillance-cameras-will-be-watching-globally-in-2021.html.

[IP21] Watchful IP. *Unauthenticated Remote Code Execution (RCE) vulnerability in Hikvision IP camera/NVR firmware (CVE-2021-36260)*. Zář. 2021. URL: https://watchfulip.github.io/2021/09/18/Hikvision-IP-Camera-Unauthenticated-RCE.html.

[Ipv17] Ipvideomarket. *Hikvision Backdoor exploit*. Zář. 2017. URL: https://ipvm.com/reports/hik-exploit.

[Ipv21] Ipvideomarket. *Hikvision has "highest level of critical vulnerability," impacting 100+ million devices*. Zář. 2021. URL: https://ipvm.com/reports/hikvision-36260.

[lib] libsodium. *Authenticated encryption*. URL: https://doc.libsodium.org/public-key_cryptography/authenticated_encryption.

[NVD] NVD. *NVD - CVE-2021-33044*. URL: https://nvd.nist.gov/vuln/detail/CVE-2021-33044.

[Pag22] Carly Page. *US government bans Huawei, ZTE and hikvision tech over 'unacceptable' spying fears*. Lis. 2022. URL: https://techcrunch.com/2022/11/28/fcc-huawei-zte-hikvision-hytera-dahua-ban.

[Pao20] William Pao. *The top 10 biggest companies in video surveillance*. Lis. 2020. URL: https://www.asmag.com/showpost/31985.aspx.

[RM22] Research a Markets. *Global surveillance camera market to 2027 with Chinese companies such as Hikvision and dahua dominating the market*. Zář. 2022. URL: https://www.globenewswire.com/news-release/2022/09/29/2524920/28124/en/Global-Surveillance-Camera-Market-to-2027-with-Chinese-Companies-Such-as-Hikvision-and-Dahua-Dominating-the-Market.html.