Raport Ştiinţific şi Tehnic

Hub inovativ pentru tehnologii avansate de securitate cibernetică (ATLAS)

Contract Nr. 17PCCDI/2018

- Etapa III -

Introducere

Etapa III s-a desfăşurat în perioada 01/01/2020 - 31/12/2020 şi a avut ca obiective principale:

- integrarea şi testarea produselor / serviciilor dezvoltate în cadrul platformei cyber range;
- achiziţia celui de-al treilea lot de echipamente;
- replicarea platformei cyber range la nivelul tuturor partenerilor şi federarea acestor platforme;
- constituirea de repository-uri cu scenarii de exercitii de apărare cibernetică la nivelul fiecărui partener;
- diseminarea rezultatelor cercetării prin publicarea de articole în jurnale şi la conferinţe şi prin intermediul site-ului Web de prezentare a proiectului (www.security-hub.ro)

Pentru atingerea acestor obiective, în cadrul celor 3 proiecte componente, au fost planificate 5 activităţi de tipul A3 (Dezvoltare experimentală). În continuare, sunt prezentate în detaliu rezultatele ştiinţifice şi tehnice obţinute în urma derulării acestor activităţi.

Proiect Component 1 - Metode formale şi tehnici de învăţare automată pentru securizarea aplicaţiilor şi sistemelor de operare

Activitatea 3.1 - Denumire Activitate: Integrarea şi testarea soluţiilor de securitate a sistemelor de operare moderne

În cadrul activităţii de cercetare în securitatea sistemelor de operare moderne am continuat lucrul, integrarea şi folosirea proiectelor din cadrul echipei Malus Security[1]. Proiectele sunt disponibile sub licenţă open source, pentru a fi folosite în evaluarea securităţii pe dispozitive iOS de alte echipe de cercetare şi dezvoltare:

- proiectul Sandblaster de inversare a profilelor de sandbox pentru iOS
- proiectul Sandscout de investigare a defectelor în profilele iOS
- proiectul iExtractor de automatizare a analizei statice pe iOS
- proiectul iOracle de analiză a controlul accesului în iOS
- proiectul Kobold de evaluează controlul accesului pentru NSXPC (mecanismul de tip IPC pentru iOS)

[1] https://github.com/malus-security/

În perioada de raportare (01/01/2020 - 31/12/2020) am urmărit integrarea şi testarea soluţiilor de securitate urmărind două scenarii:

- descoperirea de scurgeri de date private (privacy leaks) pentru aplicaţii iOS
- folosirea de profile de sandbox personalizate (custom) pe aplicaţii, pentru a permite minimizarea profilelor pe specificul aplicaţiei

Analiza exfiltrării datelor private pentru aplicaţii iOS

Una dintre cele mai importante părţi ale securităţii este legată de confidenţialitatea datelor (data privacy). Confidenţialitatea este capacitatea pe care o organizaţie sau o persoană o are de a determina ce date dintr-un sistem informatic pot fi accesate de către terţi[2]. Majoritatea ţărilor lumii au o reglementare strictă privind confidenţialitatea datelor cu privire la modul, momentul şi motivul pentru care informaţiile ar trebui distribuite între părţi. De exemplu, în Statele Unite, Legea privind protecţia confidenţialităţii online a copiilor (COPPA[3]) oferă părinţilor controlul asupra informaţiilor pe care site-urile web sau aplicaţiile mobile le pot colecta de la copiii lor. Regulamentul general privind protecţia datelor (GDPR[4]) este menit să armonizeze legislaţia privind confidenţialitatea datelor în toată Europa.

Orice aplicaţie instalată pe un dispozitiv iOS comunică cu servere externe şi partajeaza informaţii cu acestea din diferite motive. Utilizatorul este întotdeauna întrebat dacă permite aplicaţiei să partajeze anumite informaţii private cu serverul (de exemplu, în scopuri statistice), dar numai pentru acele informaţii.

Aplicaţiile de telefonie mobilă instalate pe dispozitivele Apple sunt de obicei aplicaţii de la dezvoltatori externi. Din acest motiv, toate aplicaţiile trebuie să treacă de procesul de verificare Apple înainte de a fi publicate. Nu ştim prea multe despre ceea ce face procesul de verificare pentru a valida dacă aplicaţia respectă sau nu standardele lor, deci nu ştim cum este abordată confidențialitatea datelor de către Apple.

Vrem să oferim o soluţie care poate garanta că o aplicaţie instalată pe un dispozitiv iOS nu partajeaza nicio informaţie personală. Vrem să oferim un răspuns la întrebarea de cercetare "Cât de puternic este modelul de securitate Apple iOS în prevenirea scurgerilor de confidenţialitate prin aplicaţii terţe potenţial dăunătoare?" Prin crearea unui analizor dinamic automat care va fi capabil să detecteze partajarea de date de confidențialitate făcute de orice aplicație iOS şi să ofere un răspuns la întrebarea de mai sus.

Propunem un flux de lucru care ţine de specificul iOS şi accesul la resursele sensibile ale sistemului. Există două componente principale care nu pot lipsi: un dispozitiv mobil iOS jailbroken conectat la Internet şi o aplicaţie instalată pe acesta care ar putea necesita acces la Internet. În majoritatea cazurilor, aplicaţia client comunică de obicei către serverul său prin Internet. Acest trafic este ceea ce ne interesează cel mai mult. Discutam în continuare despre cine poate fi acea entitate externă şi de ce acest lucru este destul de important.

După lansarea aplicaţiei, trebuie să generăm trafic ca şi cum ar face un utilizator obişnuit. De asemenea, trebuie să avem o altă componentă care poate intercepta şi înregistra traficul de reţea. O altă componentă separată este responsabilă pentru analiza traficului pe care îl colectăm. Vom obţine astfel un set de aplicaţii care au exfiltrat unele informaţii pe servere externe. Deoarece fiecărei aplicaţii i se poate permite să efectueze acţiuni diferite (de exemplu, una ar putea accesa locația utilizatorilor în

[2] https://searchcio.techtarget.com/definition/privacy-data-information-privacy

[3] https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule

[4] https://gdpr-info.eu

timp real, dar alta nu are voie să facă asta), le definim într-o zonă separată. Combinând zona separată cu informaţiile colectate după analizarea traficului, vom putea spune dacă o aplicaţie partajeaza date private, nu ar trebui sau nu.

Fluxul de lucru implică faptul că adunăm un set de jurnale care trebuie analizate. Considerăm că definirea unui model formal al problemei noastre uşurează acest proces. Pentru a defini acest model, trebuie să descriem cele 3 componente principale pe care le avem: datele utilizatorului, aplicaţia, serverul de la distanţă. Împărţim datele în 2 categorii separate: datele utilizatorului care sunt definite în sistem şi datele care sunt exfiltrate. Chiar dacă ultima menţionată este inclusă în prima, le tratăm ca entităţi separate în definiţia noastră.

Conform reglementărilor internaţionale, datele definite în cadrul unui sistem pot fi sau nu sensibile şi definim o premisă în acest sens, după cum urmează. Pentru a defini un model formal, trebuie mai întâi să definim ce tipuri de dovezi de susţinere avem. Acestea sunt de fapt mapări de informaţii între 2 dintre cele 4 componente din Figura 3.1.1: A (date accesibile), P (aplicaţie), E (date exfiltrate), R (serviciu la distanţă).

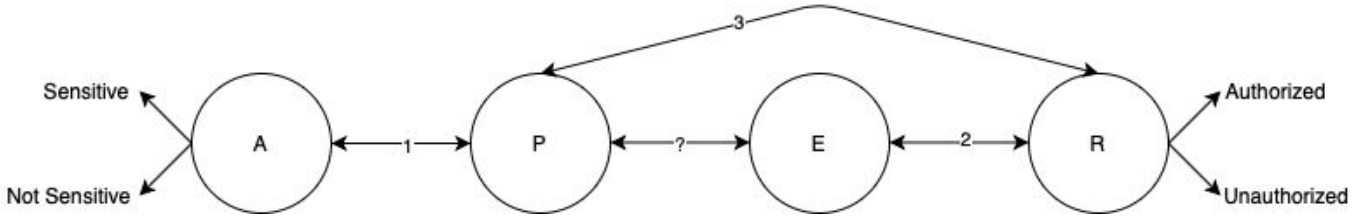

Figura 3.1.1. Modelul formal privind tipurile de date

Pentru a defini premisele legate de prima mapare, trebuie să aflăm la ce date poate accesa o aplicaţie. Acest pas este personalizat pentru fiecare aplicaţie utilizată. Un utilizator iOS are ocazia să acorde sau să refuze accesul la anumite date şi acest lucru este definit separat pentru fiecare aplicaţie instalată.

Când colectăm date despre conexiunea dintre datele exfiltrate şi serverul de la distanţă, folosim mitmproxy[5]. Mitmproxy este un server HTTPS care poate fi utilizat pentru a intercepta, inspecta, modifica şi reda traficul web, cum ar fi HTTP / 1, HTTP / 2, WebSockets sau orice alte protocoale protejate SSL / TLS. Mitmproxy are mai multe moduri de funcţionare, dar la nivel de bază acţionează doar ca un intermediar între dispozitiv şi Internet, care este capabil să intercepteze şi să citească cererile provenite de ambele părţi.

Din păcate, în acest moment nu putem spune care aplicaţie a generat o cerere dată. Fără a cunoaşte aplicaţia, nu putem satisface modelul descris anterior. Pentru a rezolva această problemă, descriem a treia mapare între aplicaţie şi serverul de la distanţă. Captura realizata de mitmproxy vine cu un timestamp pe care îl avem în vedere. Instrumentul *lsof* poate oferi informaţii cu privire la conexiunile active pe care le are un dispozitiv mobil. Avantajul este că aceste conexiuni sunt mapate la un proces specific, ceea ce înseamnă că cunoaştem aplicaţia care are acea conexiune activă. De asemenea, se poate salva momentul când conexiunea este stabilită sau închisă. În acest caz, dacă avem o cerere făcută într-un anumit moment în timp care are o anumită destinaţie, o putem asocia cu unul dintre intervalele identificate. Folosim *lsof* pentru a prelua numele aplicaţiei şi conexiunile deschise pe care le are. De asemenea, ignorăm programele de tip daemon.

Un instrument de automatizare pentru testarea interfeţei de utilizare poate evidenţia informaţii critice despre sistemul iOS sau unele vulnerabilităţi care pot fi exploatate. Crearea unui instrument capabil să automatizeze testarea UI pentru dispozitivele iOS ne oferă suportul pentru declanşa exfiltrarea unor

[5] https://mitmproxy.org/

informaţii private. O aplicaţie are nevoie de o anumită interacţiune din partea utilizatorului pentru a genera un anumit comportament ce duce la conectarea la server.

Appium[6] este o unealtă de automatizare a testelor pentru aplicaţii web native, hibride şi mobile. Rulează aplicaţii iOS, Android şi Windows utilizând protocolul WebDriver. Appium a fost ales din cauza documentaţiei sale. Mai mult, procesul de instalare este mai uşor. Dezavantajul său major este lipsa aparentă a suportului pentru aplicaţii terţe, centrul activităţii noastre.

Appium are 2 moduri de operare: unul cu GUI şi unul folosind scripturi. Din modul GUI putem face operaţiuni diferite, cum ar fi atingerea, căutarea unui şir, căutarea butoanelor, etc. Acest lucru este util dacă dorim să colectăm unele informaţii sau să facem un test rapid. Interfaţa are aceleaşi capacităţi şi utilizare pentru simulator sau dispozitiv fizic. Pentru a utiliza întreaga putere a instrumentului Appium, crearea unui script oferă o mulţime de avantaje în ceea ce priveşte testarea interfeţei de utilizare.

Appium poate testa o aplicaţie care nu este înregistrată în contul personal Apple Developer. În loc să furnizăm calea către fişierul ".ipa" putem oferi ID-ul pachetului aplicaţiei instalate pe dispozitivul fizic. Un ID pachet de aplicaţie poate fi descoperit dacă iPhone-ul este conectat la Mac.

După cum putem observa in Figura 3.1.2, Appium are un comportament similar atunci când utilizăm o aplicaţie iniţială precum Google Chrome. Putem inspecta elementele pentru a găsi câteva informaţii relevante care pot fi utilizate pentru scrierea unui script pentru automatizarea testării UI.

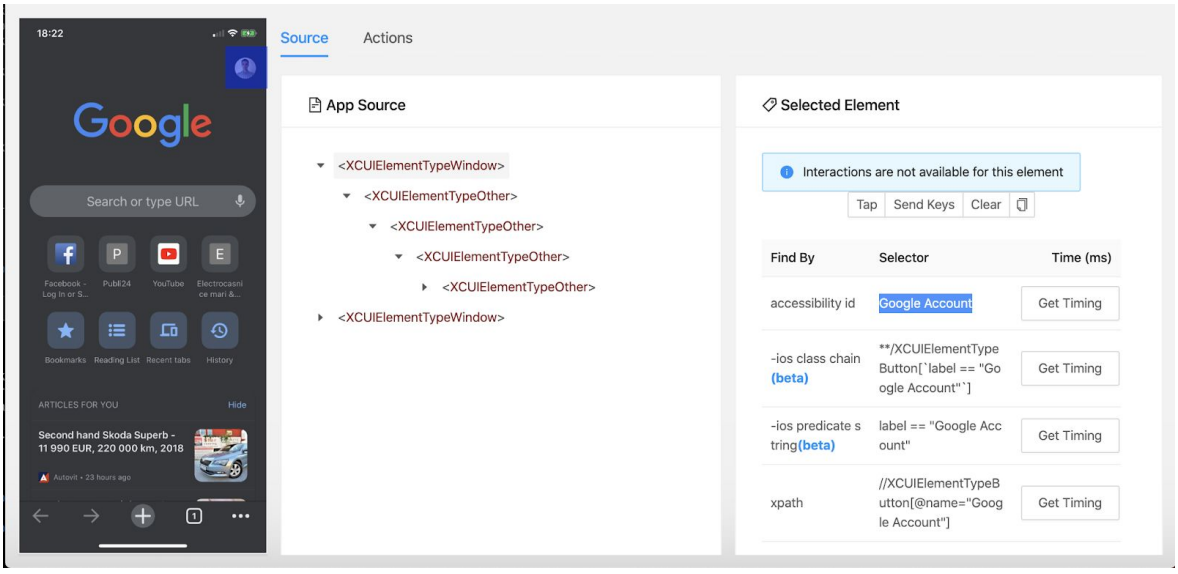

Figura 3.1.2. Interfata Appium

După colectarea unor informaţii din interfaţa Appium, putem defini scripturi pentru a testa acţiuni, cum ar fi atingerea unor butoane, căutarea unor ID-uri şi derularea. Aceste teste vor avea loc pe aplicaţii existente cu obiectivul de a declanşa exfiltrarea de informaţii private.

În prezent, avem o soluţie de investigare a scurgerilor de confidenţialitate doveditoare a conceptului, care se bazează pe munca anterioară pentru a asocia accesul aplicaţiei la resurse şi pentru a capta traficul de reţea. Dezvoltăm un instrument de automatizare UI bazat pe Appium pentru a declanşa evenimente de scurgeri de confidenţialitate. Suntem în proces de integrare a acestora pentru o soluţie complet funcţională care declanşează şi detectează scurgerile de confidenţialitate în aplicaţiile iOS.

[6] http://appium.io/docs/en/drivers/ios-uiautomation/

Profile de sandbox personalizate pentru aplicaţii iOS

În ciuda eforturilor de prevenire a ameninţărilor legate de securitate, un atacator trebuie să găsească o vulnerabilitate în sistem pentru a-şi atinge unul dintre scopurile sale: injectarea, controlul sau exfiltrarea de informaţii. Apple foloseşte un mecanism de izolare a aplicaţiei numit sandboxing pentru a limita potenţialele daune.

Apple iOS tratează fiecare aplicaţie terţă în acelaşi mod, indiferent de comportament şi cerinţe. Considerăm că această abordare este predispusă la erori, deoarece nu respectă principiul celui mai mici privilegiu. Prima analiză a Apple Sandbox a fost făcută de Dionysus Blazakis în articolul său, care este considerat un punct de referinţă pentru cercetările din acest domeniu. Mai multe defecte ale mecanismului Apple Sandbox au fost descoperite anterior cu lucrările noastre (Sandblaster, Sandscout). De asemenea, în cazul specific al aplicaţiilor Jekyll, un profil personalizat de sandbox nu ar fi permis ca sistemul să fie compromis. În prezent, nu există o alternativă la abordarea Apple Sandbox.

Prezentăm o alternativă la sandbox-ul generic Apple, care poate fi utilizat pentru a limita în mod eficient şi eficient fiecare aplicaţie terţă parte, utilizând un profil sandbox personalizat. Munca noastră poate fi folosita imediat de către utilizatorii de dispozitive jailbroken, Apple are ultimul cuvânt în cazul unei adopţii la scară largă.

O parte din munca noastră se bazează pe munca anterioară a lui Jonathan Levin, care a decompilat cu succes şi a portat pe iOS sandbox-exec. Acest instrument foloseşte apeluri de bibliotecă libsandbox.dylib pentru a configura un sandbox personalizat pentru un proces copil.

Scopul nostru este să prezentăm o alternativă la abordarea Apple Sandbox. Mai precis, ne propunem să proiectăm şi să testăm un set de metode pentru:

- Atasarea unui profil sandbox personalizat pentru o aplicaţie;
- Crearea unui profil sandbox personalizat (utilizând atât metode automate, cât şi manuale de generare a politicilor), urmând principiul celui mai mic privilegiului.

În soluţia propusă, fluxul general de lucru este rularea aplicaţiei fără sandbox şi apoi aplicare unui profil sandbox personalizat utilizând API-ul sandbox-exec. Propunem o altă abordare pentru izolarea aplicaţiilor terţe prin proiectarea unei biblioteci care oferă trei funcţionalităţi principale: restricţionarea, inspectarea şi monitorizarea aplicaţiilor.

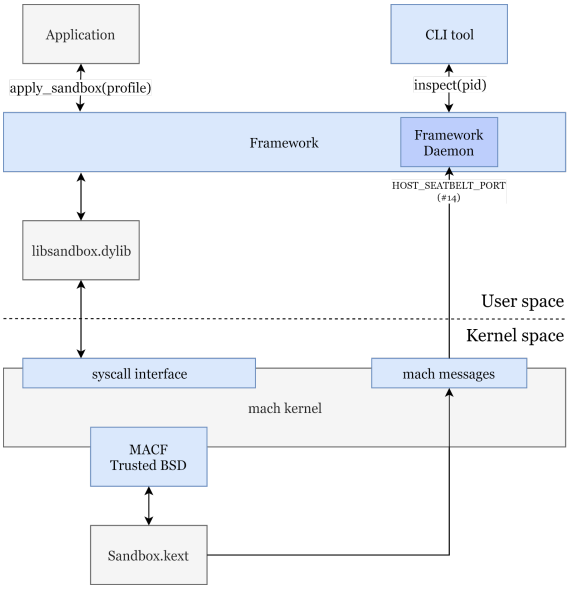

Figura 3.1.3. Aplicarea profilelor de sandbox

Modul de aplicare este funcţionalitatea bibliotecii noastre care le permite utilizatorilor să testeze o aplicaţie sau un proces. Pentru a aplica un profil sandbox vom folosi acelaşi API ca Apple şi vom folosi libsandbox.dylib.

Pe fiecare sistem de operare Apple, un proces copil moşteneşte contextul sandbox de la părintele său. De asemenea, politicile şi restricţiile sunt aplicate numai în momentul accesarii resurselor. De exemplu, dacă accesul la sistemul de fişiere este restricţionat, o aplicaţie nu va putea efectua un nou apel open(). Cu toate acestea, dacă procesul are deja un descriptor de fişier deschis cu capacităţile adecvate, îl poate folosi în condiţii normale.

Datorită acestor implicaţii, trebuie să adoptăm un design adecvat şi să începem procesul neîngrădit, să solicităm resurse speciale (dacă este necesar), apoi să testăm aplicaţia cu profilul personalizat. Pe iOS, toate profilurile sandbox ale sistemului sunt precompilate şi încorporate în imaginea firmware-ului. Apple expune un set de funcţii care acceptă profilul sandbox ca binar. Pentru a simplifica procesul, am introdus un wrapper care acceptă un profil sandbox în orice format: SBPL sau binar.

Fluxul de aplicare, presupunând că aplicaţia este iniţial fără sandbox, poate fi descris după cum urmează: (1) aplicaţia trebuie să apeleze funcţia de bibliotecă apply_sandbox(); (2) se apelează sandbox_apply_container() din libsandbox.dylib care foloseşte un apel de sistem (__sandbox_ms) pentru a aplica un profil compilat pe un proces; (3) funcţia mac_policy_register() este apelată pentru a înregistra Sandbox.kext în cadrul de control obligatoriu de acces (MACF).

Din acest moment, aplicaţia rulează într-un sandbox descris de profilul nostru şi fiecare apel de sistem nou va fi verificat de Sandbox.kext pe baza profilului sandbox furnizat. Extensia kernel Sandbox acţionează ca un modul de politică MACF şi analizează cererea de operaţie. Reţineţi că toate modulele de politică înregistrate trebuie să aprobe cererea de operare pentru a permite apelul de sistem.

Pentru a verifica dacă un profil de sandbox defineşte mediul de sandbox dorit, am introdus modul de inspecţie. La fel ca aplicarea, interacţionează cu libsandbox.dylib pentru a inspecta politicile de securitate aplicate şi pentru a identifica ce profil descrie sandbox-ul curent al unui proces dat. Implementarea noastră pentru acest mod este puternic inspirată de sbtool-ul lui Jonathan Levin[7] şi expune informaţii despre mediul curent sandbox, cum ar fi: numele profilului sandbox, dimensiunea , numărul de reguli, extensii şi multe altele.

Modul de monitorizare poate fi utilizat pentru a colecta informaţii despre încercările de încălcare a politicilor. De asemenea, poate fi utilizat pentru a genera profiluri personalizate de sandbox care reflectă comportamentul aplicaţiei pentru o anumită perioadă de timp. Se recomandă utilizarea acestui mod într-un mediu controlat pentru a limita şansa de a surprinde un comportament malign.

O soluţie pentru proiectarea modului de monitorizare este de a construi un profil special sandbox care conţine operaţiunea de monitorizare şi de a-l aplica. Prezenţa operaţiei de monitorizare configurează extensia kernel-ului Sandbox în modul de monitorizare pentru procesul curent. În acest mod, MACF permite toate apelurile de sistem şi înregistrează fiecare încălcare a profilului sandbox prin trimiterea unui mesaj mach. Aceste mesaje pot fi capturate în spaţiul utilizatorului.

O altă soluţie este să configuram extensia kernel Sandbox pentru a înregistra toate mesajele pentru toate nivelurile de înregistrare, apoi pentru a colecta regulile şi a construi un profil. În mod implicit, Sandbox.kext înregistrează numai evenimentele de încălcare a regulilor. Am testat această metodă atât

[7] http://newosxbook.com/src.jl?tree=listings&file=sbtool.c

pe platformele macOS, cât şi pe cele iOS şi am reuşit să construim un profil sandbox care să se potrivească unei sesiuni de rulare specifice pentru o aplicaţie terţă parte.

În implementarea noastră am luat în considerare a doua opţiune datorită caracteristicilor sale de disponibilitate şi flexibilitate. Metodologia noastră poate fi utilizată pentru a profila un proces şi a identifica care sunt resursele care sunt accesibile printr-o verificare a sandbox-ului, pentru a aplica o politică de securitate descrisă de un profil de configurare numit sandbox profile şi pentru a verifica dacă procesul a fost izolat cu succes.

În continuare lucrăm la îmbunătăţirea proiectelor şi la descoperirea de noi scenarii de utilizare:

- oferirea suportului de iOS 13 şi iOS 14 pentru SandBlaster
- dezvoltarea completă a soluţiei de UI exerciser pentru iOS
- validarea profilelor de sandbox obţine cu ajutorul SandBlaster
- minimizarea profilelor de sandbox pentru specificul unei aplicaţii iOS
- definitivarea investigaţiei de scurgeri de date private (privacy leaks) pentru aplicaţii iOS

Activitatea 3.2 - Denumire Activitate: Integrarea şi testarea soluţiilor de metode formale şi învăţare automată pentru securitate

În etapa finală a proiectului a fost integrată şi testată soluţia pentru securitate folosind metode formale şi învăţare automată. Avand in vedere că soluţia de securitate a avut doua componente distincte (RIVER şi MALID), le vom aborda pe fiecare separat.

RIVER şi metode de AI pentru fuzz testing

Pentru RIVER (disponibil online la https://river.cs.unibuc.ro), am creat o aplicaţie de tip CLI (Command Line Interface) prin care putem observa în timp real defectele de securitate găsite de RIVER la rularea unei aplicaţii, dar şi salvarea în directoare a intrărilor care au cauzat problemele, separate pe tipurile de erori gasite. Astfel, framework-ul nostru poate fi utilizat de dezvoltatorii de software pentru a-şi testa codul înainte de a-l lansa în producţie. De asemenea, am implementat o nouă metoda de a alege mai bine intrările astfel incat folosim tehnici de fuzzing doar pe părţile din intrare care sunt folosite efectiv de program (analiza de tip taint), obtinand astfel o mai buna acoperire a codului într-un timp mai mic. Multiplele experimente bazate au arătat ca metodele dezvoltate sunt mai eficiente atunci cand sunt augmentate cu metode de inteligentă artificială. Rezultatele au fost validate de comunitatea ştiinţifică prin acceptarea la conferinţe şi workshopuri de specialitate. Momentan finalizam ultimele detalii componenta RIVER pentru a o putea înscrie în comparatorul de tool-uri de fuzz testing organizat de către Google: https://google.github.io/fuzzbench

În continuare vom prezenta diverse teste aplicate asupra frameworkului RIVER. O parte din ele au fost raportate in articolele [19, 10, 14] publicate de echipa UB in 2020.

În lucrarea [10] este prezentată o nouă metoda de testarea a fişierelor binare folosind algoritmi de tip Reinforcement Learning (RL) pentru a reduce spaţiul căutării şi a obţine acoperirea codului mai rapid decat dacă folosim algoritmi clasici de tip concolic execution. Algoritmul consta în învăţarea unor modele pentru a prezice într-un mod eficient care sunt condiţiile care trebuie modificate în program pentru a găsi defecte în codul sursa. Arhitectura de baza a algoritmului este un model de tip Deep Q-Network care foloseşte celule LSTM pentru a genera noi intrării în timpul testarii.

Testarea aplicatiei RIVER a fost realizata pe doua aplicaţii open-source de tip parser: JSON (https://github.com/nlohmann/json) si HTTP (https://github.com/nodejs/http-parser), ambele aplicatii fiind implementate in C++. Pe langa folosirea codului au mai trebuit adaugate programe de tip wrapper pentru a putea realiza comunicarea între aplicaţiile şi framework-ul RIVER.

Model	HTTP parser	JSON parser
RiverConcolic	2h:10m	2h:53m
RiverConcolicRL	1h:37m	2h:14m

Figura 3.2.1

În imaginea de mai sus (Figura 3.2.1) putem observa rezultatele comparative pentru a găsi 100 de basic block-uri diferite atunci când folosim metoda default (RIVERConcolic) în comparaţie cu algoritmul de Reinforcement Learning (RL) pe cele două programe de test. Se poate observa ca noul model poate atinge aceeaşi performanţă mult mai repede (cu 33 minute pentru parser-ul HTTP şi cu 37 de minute pentru parser-ul JSON).

De asemenea, am testat dacă atunci cand apar mici modificări ale codului (între 10 şi 50 de linii de cod) şi folosim modelul invatat prin RL, framework-ul îşi pastreaza performanţele obţine (Figura 3.2.2).

Model	HTTP parser	JSON parser
RiverConcolic	1h:56m	2h:47m
RiverConcolicRL	1h:31m	2h:15m

Figura 3.2.2

În acest caz, observam ca daca refolosim testele încă obţine o îmbunătăţire de 27% pentru parser-ul HTTP, respectiv 24% pentru parser-ul JSON, în comparaţie cu algoritmul fără RL.

Totodată, am vrut să testăm dacă folosind *învăţarea* pe versiuni mai vechi ale codului şi *testarea* pe versiuni mai noi, obţinem rezultate satisfăcătoare. Motivul acestui test este de a verifica învăţarea de tip online, ceea ce înseamnă ca testăm adaptabilitatea algoritmului. Pentru aceasta varianta, modelul a fost antrenat 24 de ore pe codul-sursă vechi şi o ora pe cel nou (diferenta de timp între cele doua versiuni ale codului-sursă a fost de un an).

Model	HTTP parser	JSON parser
RiverConcolic	2h:05m	2h:38m
RiverConcolicRL	1h:39m	2h:07m

Figura 3.2.3

Şi de aceasta data observam ca timpul obtinut de algoritmul de tip RL a fost mai scurt decat versiunea clasică a algoritmului.

În lucrarea [14] este prezentată pe larg partea de Concolic Execution Engine din cadrul framework-ului RIVER. Chiar dacă exista un algoritm similar cu RIVER - Microsoft SAGE, acesta nu are codul sursa facut public. Contribuţia noastra reprezinta implementarea open-source si optimizarea aceastui algoritm, precum şi prezentarea problemelor şi bottleneck-urilor care au apărut în timpul dezvoltării programului.

Evaluarea a fost realizata pe trei aplicaţii open-source de tip parser: JSON, HTTP si XML (http://xmlsoft.org) pe un PC avand procesor Intel i7, cu 6 core-uri, 16 GB RAM şi placă video NVidia RTX 2070 pe Ubuntu 16.04.

Metodele de evaluare au fost aceleaşi ca cele folosite de Microsoft SAGE prezentate în lucrarea “SAGE: Whitebox fuzzing for security testing”, dar implementate pentru framework-ul nostru, din moment ce sursele pentru SAGE nu au fost publice.

Pentru aceasta evaluare am rulat cele doua modele de evaluare existente care foloseau componenta Concolic Execution Engine. Primul model foloseşte un algoritm clasic genetic şi dynamic taint analysis

prezentat în “Fuzz Testing with Dynamic Taint Analysis based Tools for Faster Code Coverage”, care are performanţe asemănătoare cu toolul AFL pe testele efectuate. Cel de-al doilea model foloseşte componenta “Concolic Execution Engine” implementata in RIVER (RiverConc).

Model	30m	1h	2h	3h
River fuzzing	621	765	783	815
RiverConc	291	402	809	863

Figura 3.2.4

În tabelul din Figura 3.2.4 putem observa cum variază pentru programul de test “XML parser” numărul de basic block-uri găsite în funcţie de timpul de rulare al programului (prima linie) şi de algoritmul folosit (prima coloana). Astfel, algoritmul genetic găseşte într-un timp scurt multe basic block-uri (621, comparativ cu 291 şi 765 comparativ cu 402), dar pe masura ce lăsăm programul sa meargă, observam ca RiverConc obţine rezultate mai bune pentru programul de test.

Model	30m	1h	2h	3h
River fuzzing	73	89	97	111
RiverConc	28	52	88	109

Model	30m	1h	2h	3h
River fuzzing	50	79	94	97
RiverConc	32	61	90	104

Figura 3.2.5

Testăm în acelaşi mod RIVER, dar pentru un alte 2 programe de test - “HTTP parser” si respectiv “JSON Parser” şi obţinem rezultate asemănătoare cu tabelul anterior, cf. tabelul din stanga, respectiv dreapta din Figura 3.2.5.

Considerăm că putem realiza un compromis între cei doi algoritmi folosiţi. Dacă se doreşte o căutare rapidă a defectelor, se poate folosi algoritmul genetic. În schimb dacă produsul este critic pentru securitatea sistemelor se poate rula o perioada mai mare timp cel de-al doilea algoritm sau se pot rula algoritmii simultan.

Aceste rezultate explica intuiţia iniţială care ne spune ca versiunea care foloseşte componenta “Concolic Execution Engine” găseşte mai greu basic block-urile deoarece apelează o librărie externă, Microsoft Z3, în rol de SMT solver care consuma multe resurse.

În ciuda acestui fapt (folosirea SMT solver-ului care duce la o performanţa scăzută a sistemului) dacă rulam programul suficient timp putem avea o performanţă mai bună din punctul de vedere al găsirii defectelor din moment ce acest algoritm găseşte exact intrările care cauzează probleme în loc sa incerce aleator mai multe intrări şi să obţină pe aceasta cale acoperirea codului sursa.

În lucrarea [19] este prezentată o a doua iteraţie asupra framework-ului RIVER 2.0, care oferă o arhitectura imbunatatita. Articolul prezentate în detaliu arhitectura şi toate componentele RIVER (Taint Analysis, Online Guided Fuzzer, Offline Guided Fuzzer si Concolic Execution Engine).

Pentru testarea acestei noi versiuni am prezentat în tabelul din Figura 3.2.6 timpul de antrenare pentru mai multe epoci pentru algoritmul de învăţare automată, care poate genera fişiere HTTP, XML si PDF noi plecând de la un set iniţial de fişiere date.

Num epochs	HTTP	XML	PDF objects
50	8h:25	7h:19	9h:11
40	6h:59	5h:56	8h:04
30	5h:35	4h:20	6h:15
20	3h:48	3h:42	4h:17
10	2h:10	1h:12	3h:02

Figura 3.2.6

Din acest tabel putem observa că învăţarea a fost mai rapidă în cazul fişierelor XML si request-urilor HTTP in comparaţie cu fişierele PDF, dar acest lucru se poate explica prin faptul ca structurile fişierelor XML si HTTP sunt mai simple în comparatie cu structura fişierelor PDF.

File type	Time in minutes
XML	49
HTTP	25
PDF	51

Figura 3.2.7

În tabelul din Figura 3.2.7, putem observa timpii necesari producerii a 10.000 de noi fişiere PDF şi XML şi a 50.000 de noi requesturi HTTP. Motivul principal pentru discrepanţa dintre cele doua tipuri de generari (ale fisierelor, respectiv request-urilor) este data de dimensiunea obiectelor (request-urile avand o dimensiune redusă).

Model	9h	15h	24h	72h
HTTP-fuzz+genetic	229	230	230	232
HTTP-Sample	238	249	257	271
HTTP-SampleSpace	241	245	269	279

Figura 3.2.8

În cel de-al treilea tabel, din Figura 3.2.8, avem rezultatele comparative între trei algoritmi de fuzzing implementaţi în RIVER. Putem observa ca şi după 24 de ore, încă găsim noi basic block-uri, care pot duce la găsirea de defecte în cod.

În momentul de faţă dorim sa finalizam o testare cat mai completa a RIVER in platforma automată de testare Google FuzzBench. După ce RIVER va implementa toate conexiunile cerute de FuzzBench, vom putea vedea rezultate comparative cu alte soluţii existente de fuzzing ca AFL, AFL++, KLEE şi libFuzzer. Testarea va fi făcută de Google pe 24 de aplicaţii reale, pe o perioada de 24 de ore, iar rezultatele vor fi făcute publice şi se vor putea descărca în format CSV. După ce vom avea aceasta validare finală într-un mediu industrial (creat de Google), pe langa publicarea unui articol într-o conferinţa sau jurnal academic, vom promova RIVER în randul companiilor de securitate.

MALID si metode de dictionary learning

Pentru a doua componenta de securitate dezvoltate la UB, numita MALID, aria principală de cercetare a fost investigarea folosirii tehnicilor de Dictionary Learning (DL) pentru detectarea aplicaţiilor de tip malware. Deoarece domeniului îi lipseşte standardizarea în ceea ce priveşte algoritmii de învăţare automată pentru detecţia malware, precum şi în ceea ce priveşte bazele de date malware, studiile iniţiale au identificat seturi de date (benchmark-uri) de programe de tip malware pentru experimentare. Astfel într-o prima faza am identificat 2 baze de date potrivite pentru aplicaţia noastră: EMBER, o colecţie de 1.1 milioane fişiere portable executable şi DREBIN, un set de date conţinând aplicaţii Android. Scopul testelor este acela de a aplica clasificarea DL în problema identificării malware-ului, o abordare care, după cunoştinţele noastre, nu a mai fost încercată înainte şi de a testa performanţa algoritmilor în identificarea mai multor clase de familii malware. O abordare nesupravegheată (unsupervised learning) poate fi, de asemenea, testată, unde adevărata clasă malware este necunoscută, verificând astfel adecvarea metodelor DL pentru a descoperi structura problemei de detectare de malware.

Implementările noastre au fost testate pe cele două baze de date disponibile pe internet (public benchmarks) menţionate anterior şi comparate cu un alt algoritm nesupervizat de antrenare online de dicţionare, LC-RLSDLA, care însă selectează semnalele ce participă la reantrenare, fiind deci mai puţin potrivit pentru detecţia de malware. De asemenea, am comparat performanţele cu un al doilea algoritm

de antrenare online de dicţionare, OSSDL. Varianta nesupervizată a acestui algoritm nu se pretează la contextul aplicaţiei, deoarece la sfârşitul antrenării clasificatorul este calculat doar folosind fişierele adnotate, ca atare nu poate fi utilizat în faza online.

Rezultatele, care au fost detaliate în articole ştiinţifice, arată că pe baza de date de aplicaţii Android (DREBIN), metoda propusă are performanţe mai bune decât ceilalţi algoritmi. Pe baza de date de fişiere Windows (EMBER), metoda noastră are performanţe similare cu algoritmul LC-RLSDLA şi mai bune decât OSSDL. Reamintim avantajul faţă de metoda LC-RLSDLA pentru contextul specific al aplicaţiilor de securitate.

După acest mic succes de etapa, am investigat în continuare posibilităţile de utilizare a metodei de învăţare automată numită Dictionary Learning (DL) pentru detecţie nesupervizată de anomalii. În această clasă de probleme se înscrie şi problema identificării malware. Necesitatea unor metode nesupervizate este motivată de faptul că în astfel de aplicaţii numărul anomaliilor este extrem de redus în comparaţie cu fişierele legitime şi adesea bazele de date nu conţin toate tipurile de anomalii existente. În plus, în cazul malware, noi forme de aplicaţii malware sunt dezvoltate în permanenţă, fiind necesari algoritmi care să permită identificarea lor fără a beneficia de exemple în prealabil.

In [16] am detaliat un prim set de experimente care a constat în extinderea testelor unui algoritm (TODDLeR) dezvoltat în anul precedent în cadrul proiectului [2] la alte baze de date. Algoritmul este o metodă semisupervizată, online, de detecţie de anomalii. Testele au vizat baza de date de fraude cu cardul, dezvoltată de Universite Libre de Bruxelles şi au fost efectuate pe un subset de N = 39.754 de tranzacţii. În urma verificării mai multor variante de parametri, rezultatele cele mai bune au fost de 98.86% acurateţe de clasificare şi rata fals pozitivă 83%.

A doua metodă dezvoltată este complet nesupervizată şi presupune tehnici de filtrare progresivă a semnalelor după o serie de criterii ce restrâng baza de date la un set de candidaţi - anomalii. Unul din aceste criterii presupune ca procentul aproximativ de anomalii din baza de date să fie cunoscut, fapt ce se întâmplă adesea în practică. Unul din obiectivele principale ale algoritmului este reducerea numărului de negative false. În acest sens, metoda, aplicată pe aceeaşi bază de date de mai sus, a returnat 0 negative false concomitent cu reducerea setului de posibile anomalii la jumătate din baza de date.

Cea de-a doua direcţie de cercetare, detaliată in [12], se referă la metode de optimizare pentru funcţii obiectiv composite, ce conţin elemente stocastice nenetede. Algoritmul dezvoltat reprezintă o variantă minibatch pentru algoritmul Stochastic Proximal Gradient (SPD). Ideea de bază a adaptărilor de tip minibatch constă în aceea că în fiecare iteraţie SPD, se calculează media gradienţilor pe un număr limitat de semnale, iar pasul se execută în direcţia negativă. Alegerea dimensiunii minibatch-urilor este esenţială, deoarece un număr mai mare de semnale presupun o varianţă mai mică a direcţiei, ce se traduce într-o viteză mai bună de convergenţă. Un număr prea mare de semnale, însă, înrăutăţeşte performanţele. Lucrarea conţine o analiză detaliată a complexităţii numerice a soluţiei. Metoda a fost testată de asemenea pe o problemă de clasificare, anume o aplicaţie Support Vector Machine (SVM).

Avand in vedere ca testele aplicate asupra MALID au fost promiţătoare, vom continua sa optimizam aplicaţia pentru a o pregăti pentru a fi folosită în industrie.

Proiect Component 2 - Arhitecturi de securitate pentru IoT

Activitatea 3.3 - Integrarea soluţiilor IoT în cadrul platformei cyber range

Pentru a testa eficienţa arhitecturii propuse în etapa anterioară şi pentru a putea analiza comportamentul platformei cyber range în cadrul unui scenariu real, cu componente active şi bazat pe o arhitectură de referinţă pentru smart home şi smart city, a fost implementată o soluţie Internet of Things (IoT), *IoT4mSCp*, ce a avut ca obiectiv monitorizeze nivelurile de poluare, în special nivelurile prezente

în oraşe, prin utilizarea mai multor dispozitive IoT şi senzori ataşaţi ce transmit date la o serie de servicii ce au rulat pe platforma cyber range.

Scopul acestei soluţii este de a integra în platforma cyber range o soluţie reală care acoperă pe toate nivelurile o arhitectură IoT, de la senzori până la serviciile software care rulează în backend:

- Nivelul senzorilor care înregistrează date şi transmit date
- Nivelul comunicării dintre nodurile IoT şi serviciile din backend
- Nivelul serviciilor software care rulează în backend

Prin instalarea acestei soluţii, este realizată o testare completă a platformei cyber range şi soluţia IoT devine pentru utilizatorii platformei:

- Un studiu de caz pentru soluţie verticală în domeniul IoT;
- Un scenariu utilizat pentru evaluarea securităţii unei soluţii IoT; pe baza acestuia pot fi definite scenarii de tip atac asupra unor componente ale arhitecturii;
- Un scenariu utilizat pentru evaluarea vulnerabilităţilor unei soluţii IoT complete, precu şi pentru a testa măsuri care să contribuie la creşterea securităţii.

Ideea soluţiei *IoT4mSCp* a stat la baza arhitecturii IoT definită în etapa din 2019 a proiectului şi presupune plasarea mai multor staţii în mai multe zone ale oraşelor. Aceste staţii încarcă periodic şi trimit date în Cloud-ul IoT, format din serviciile ce rulează în platforma cyber range. Pentru testarea conceptului şi a prototipului, au folosit plăci de dezvoltare, precum Raspberry Pi, dar pentru soluţia IoT finală pot fi folosiţi senzori de înaltă calitate şi dispozitive industriale de tip IoT gateway, precum HMS Netbiter sau echipamente echivalente. Diferenţa semnificativă între această soluţie şi alte soluţii similare este dată de importanţa acordată securităţii gateway-ul IoT şi a comunicaţiile către Cloud-urile IoT, cloud ce a fost replicat în platforma cyber range a ASE.

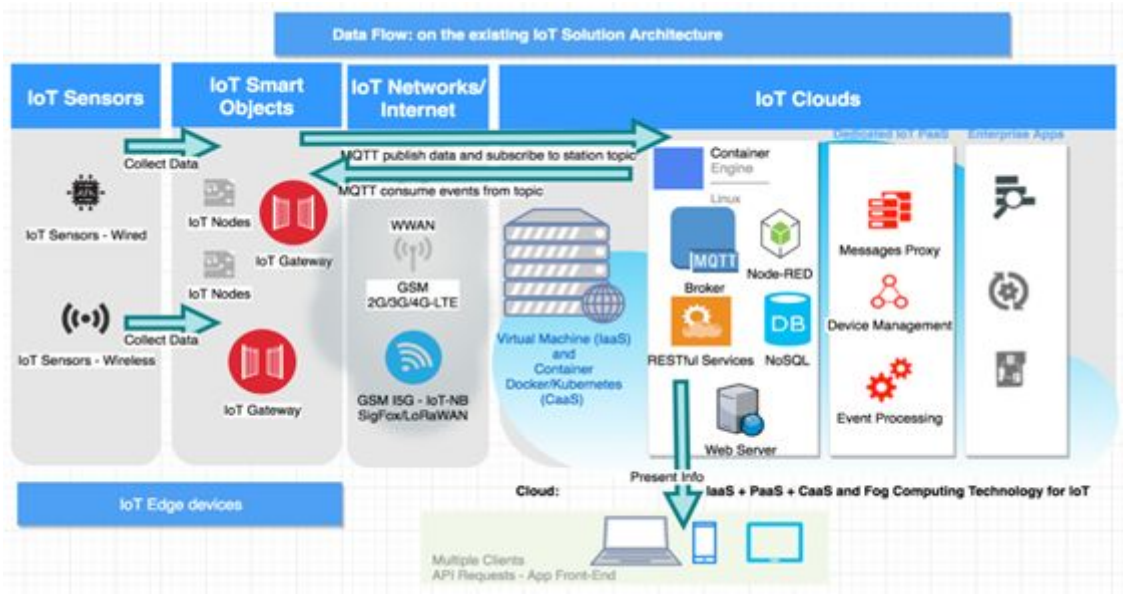

Figura 3.3.1. Arhitectură IoT cu flux de date definită în etapa 2019

Scopul soluţiei *IoT4mSCp* este de:

- a testa o arhitectură reala de tip IoT ce se regăseşte în numeroase soluţii comerciale sau open source;
- a testa platforma cyber range cu o soluţie care să acopere toate perspectivele unei arhitecturi IoT;
- a pune la dispoziţia practicienilor din domeniul securităţii cibernetice o soluţie IoT care poate fi monitorizată şi evaluată.
- a colecta, analiza şi prelucra datele primite de la senzori pentru a obţine monitorizarea în timp real a calităţii aerului, precum şi pentru a determina securitatea şi coerenţa datelor colectate.

Diagrama fluxului de date, descrisă în Figura 3.3.1, prezintă fluxul logic al prelucrării datelor şi a funcţiilor de transmitere prin arhitectura curentă. Arhitectura, definită în etapa din 2019, are următoarele componente:

Dispozitive IoT Edge:

- Receptorul IoT (circuitul inter-integrat (I2C), interfaţa serială periferică (SPI) şi receptorul / emiţătorul asincron universal (UART)) şi senzorii wireless (ZigBee / Z-Wave) utilizaţi pentru a colecta date folosind rate de eşantion: temperatură (C°), presiune (hPa), altitudine (m), nivel de umiditate relativă (%), dioxid de carbon (CO2 ppm), monoxid de carbon (CO ppm), amoniu (NH4 ppm), metan (CH4 pbm), reţele Wi-Fi detectate şi intensitatea semnalului lor în decibeli.

Obiecte IoT Smart:

- Noduri IoT - pentru prototip şi validarea conceptelor au fost folosite ESP8266, dar în producţie ar trebui utilizate unităţi de calcul MCU-uri industriale.
- Gateway-uri IoT - pentru validarea conceptelor au fost folosite plăci de dezvoltare precum Raspberry Pi 3 şi Nitrogen iMX6.

Reţelele IoT / internet - comunicaţiile către Internet / cloud IoT sunt importante şi trebuie să fie fiabile şi sigure, asigurând o acoperire minimă în oraş. Pot fi utilizate următoarele comunicaţii de reţea: GSM (2G / 3G / 4G-LTE / 5G — IoT-NB), SigFox, LoRaWAN şi Wi-Fi. Pentru această implementare au fost folosite reţele Ethernet, GSM şi Wi-Fi.

Soluţii Cloud/Backend:

- În platforma cyber range a fost instalată o maşină virtuală într-un container Docker / Kubernetes, în care au fost implementate:
    - WebServer (Apache Tomcat 9.0) cu servicii RESTful, http://tomcat.apache.org/
    - Brokerul MQTT (Mosquitto), https://mqtt.org/
    - Node.js 9+, https://nodejs.org/en/
    - Framework-uri Node-RED, https://nodered.org/
    - Baza de date NoSQL (MongoDB), https://www.mongodb.com/

- Cloud dedicat IoT PaaS (cum ar fi Google IoT, Amazon Web Services IoT, IBM Bluemix IoT şi Oracle IoT) este folosit pentru a nu depinde de niciun furnizor de cloud. Aceste sisteme de cloud sunt folosite pentru analizele dedicate seriei de timp IoT.

Sisteme Enterprise:

- Business Intelligence (BI), inteligenţă artificială (AI) şi cloud analytics de date sau sisteme enterprise pot fi utilizate pentru a oferi o interpretare a datelor colectate.

Front-end pentru gestionarea dispozitivelor şi raportarea datelor colectate (de exemplu, crearea hărţii inteligente a poluării oraşelor). Pentru această soluţie a fost dezvoltată o soluţie Web, folosind Angular, ce rulează în platforma cyber range şi care permite vizualizarea datelor colectate.

În faza de testare a platformei cyber range, prin implementarea arhitecturii IoT descrise, au fost acoperite în această fază:

- Componentele IoT receptoare, IoT Edge;
- Componentele IoT Smart;
- Reţele GSM şi Wi-Fi;
- Soluţia Cloud bazată pe maşina virtuală ce a rulat în platforma cyber range.

Posturile de monitorizare a poluării, IoT gateway, sunt abonate la brokerul MQTT prin canale/subiecte de tipul : „/ station” şi „/ station / {identifier station}”, unde {identifier station} reprezintă MAC-ul cardului de interfaţă de reţea a staţiei pentru conexiunile Wi-Fi/Ethernet/GSM utilizate. După abonarea la canalul MQTT, placa IoT aşteaptă mesaje de configurare. După iniţializare, staţiile colectează date de la

senzori la un anumit interval şi publică un mesaj MQTT către brokerul Mosquitto găzduit pe partea de server.

Componenta Node-RED persistă datele prin protocolul MQTT. Acesta lansează un flux care se abonează la mesajele publicate pe subiecte/sesiuni, preia conţinutul mesajului, îl validează şi apoi îl stochează în MongoDB.

Componenta backend *IoTP4mSCp* oferă servicii RESTful pentru interacţiunea cu baza de date MongoDB (operaţii CRUD) şi cu brokerul MQTT. Mesajele, care sunt publicate în canalele MQTT de mai sus, modifică parametrii de configurare ai unei staţii - activează / dezactivează staţia, defineşte intervalul dintre înregistrări.

Pentru a monitoriza nivelul de poluare a aerului şi a calităţii aerului, o placă de dezvoltare (cum ar fi Raspberry Pi sau Nitrogen iMX) înregistrează următoarele date prin intermediul senzorilor ataşaţi: temperatură (Co), presiune atmosferică (hPa), altitudine (m), nivel umiditate (%), nivel de dioxid de carbon (CO2 ppm), monoxid de carbon (CO ppm), metan (CH4 pbm) şi amoniu (NH4 ppm). De asemenea, sunt înregistrate date despre reţelele WI-FI identificate în apropierea staţiei şi nivelul semnalului acestora. Componentele care compun o staţie sunt:

- Placa de dezvoltare (pentru validarea soluţiei conceptuale IoT Gateway) Raspberry Pi 3 Model B.
- MCP 3008 este un convertor analog-digital, cu ajutorul căruia poate fi acceptată o dimensiune analogică la intrare şi este emis un număr reprezentând o aproximare a semnalului analogic al semnalului de intrare.
- SNS-MQ135 este un senzor electrochimic care măsoară calitatea aerului şi este capabil să detecteze niveluri de dioxid de carbon (CO2), amoniu (NH4), etanol (C2H6O) şi toluen (C7H8). Senzorul are o ieşire analogică şi digitală.
- MQ9 este un senzor de gaz care măsoară nivelul de monoxid de carbon (CO) şi metan (CH4). La fel ca SNS-MQ135, acest senzor are o ieşire analogică şi digitală.
- SYH-2R este un senzor analogic pentru detectarea nivelurilor de umiditate relativă.
- BMP280 este un senzor proiectat de Bosch pentru a detecta temperatura (precizia ± 1 ° C) şi presiunea barometrică (precizia de 1 hPa).

Fritzing (https://fritzing.org/) este un produs software open-source utilizat pentru a uşura conexiunea dispozitivelor şi circuitelor electronice. Permite proiectarea diagramelor de circuite, precum şi proiectarea plăcilor de circuite imprimate (PCB). Schema şi diagrama de conexiune au fost proiectate în Fritzing. La asamblarea componentelor, s-au utilizat firele de legătură de tip mamă-tată şi placa de conectare.

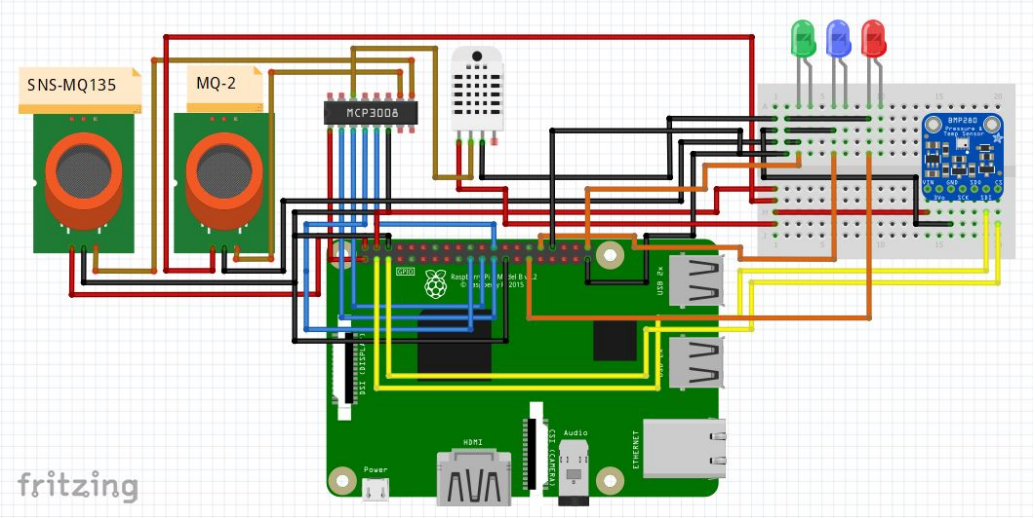

Figura 3.3.2. Diagrama hardware

Colectarea datelor de către senzorii ataşaţi se realizează atât prin interfaţa SPI (pentru comunicarea cu senzorii analogici), cât şi prin interfaţa I2C (pentru comunicarea cu senzorii digitali). Pentru a citi datele de la senzori prin I2C, a fost utilizată biblioteca DIO (Input / Output Device Java) (https://openjdk.java.net/projects/dio/). Bosch, producătorul senzorului BMP280, oferă o implementare Java pentru a prelua datele. Pentru a citi datele de la senzori prin SPI, a fost nevoie de o modalitate de a interacţiona cu convertorul analog-digital MCP3008.

Aplicaţia din gateway-ul IoT comunică datele înregistrate folosind protocolul MQTT cu implementarea sa de la Eclipse Foundation - Mosquitto MQTT. Aceasta implică existenţa unui broker care are rolul de a selecta şi filtra mesajele primite. Soluţia utilizează serviciile din backend ce găzduieşte brokerul MQTT şi aplicaţia web de gestionare/monitorizare a staţiei.

În ceea ce priveşte securitatea, firewall-ul poate fi configurat pentru a filtra conexiunile pe baza de liste de acces, whitelist, în care sunt incluse adresele IP permise. Astfel, porturile specifice pentru anumite adrese IP pot fi restricţionate. Pentru această instanţă firewall-ul este configurat pentru a permite porturile de intrare 22 - pentru SSH — Secure Shell, 8080 - să acceseze serverul web al soluţiei şi 1883 - pentru brokerul Mosquitto MQTT.

Stocarea datelor colectate de la gateway-ul *IoT4mSCp* IoT se realizează folosind o bază de date Cloud (NoSQL) MongoDB SQL. Utilizarea unei baze de date NoSQL în cadrul acestui proiect este ideală pentru stocarea eşantioanelor înregistrate de staţii şi procesarea acestora pentru a fi distribuite prin MapReduce.

MongoDB este o bază de date non-relaţională, open-source, dezvoltată în C++ şi lansată public în 2009. MongoDB este orientată spre documente, oferind performanţe ridicate şi scalabilitate. Spre deosebire de o bază de date relaţională, în care datele sunt stocate în tabele, în MongoDB, datele sunt stocate ca documente de notare JavaScript (JSON) din colecţii. Aceste colecţii nu impun o anumită schemă, deci documentele au o schemă dinamică şi pot avea o structură diferită.

Stocarea datelor înregistrate de staţiile din baza se realizează prin două colecţii:

- *staţii* - în care datele sunt stocate pe staţii (numele lor, identificatorul unic, poziţia geografică)
- *inregistrari*, o colecţie în care este asigurată persistenţa datelor transmise de senzorii IoT (nivelul calităţii aerului, reţelele Wi-Fi detectate şi semnalul acestora).

De asemenea, soluţia *IoTP4mSCp* are o interfeţe Web, ceea ce implică existenţa utilizatorilor care administrează staţiile. Aceste date sunt stocate în format JSON în colecţia destinată utilizatorilor.

Configuraţia Brokerului MQTT Mosquitto Broker a implicat instalarea brokerului pe instanţa cyber range. Toate cererile de abonare şi publicare nespecificate sunt restricţionate prin setarea „ allow_anonymous” la valoarea „false”.

Node-RED este un instrument de dezvoltare bazat pe flux dezvoltat iniţial de IBM pentru conectarea dispozitivelor hardware, a API-urilor şi a serviciilor online, ca parte a IoT. Este construit pe Node.js şi se bazează pe evenimente, ceea ce îl face ideal pentru rularea pe dispozitive de tip limited-cloud sau bazate pe cloud. Oferă un editor de fluxuri ce rulează într-un browser pe portul 1880 şi poate conecta o mulţime de funcţii de intrare, ieşire, stocare şi JavaScript. Fluxurile pot fi importate / exportate în format JSON (JavaScript Syntax Object Notation) (e.g., {“software”:”Node-RED”, “version”:1}).

Folosind o instanţă Node-RED (aceasta poate fi implementată oriunde, chiar şi pe localhost) şi un flux de dezvoltare, mesajele publicate prin MQTT de către staţii pot fi stocate în instanţa MongoDB. Acest flux conţine următoarele noduri:

1. nodul de ieşire MQTT (conectat la brokerul MQTT mosquitto);
2. nodul funcţiei JavaScript (colectează şi validează mesajul MQTT);
3. nodul de intrare MongoDb (conectat la instanţa MongoDb);
4. nodul de debug (înregistrează mesajul MQTT brut).

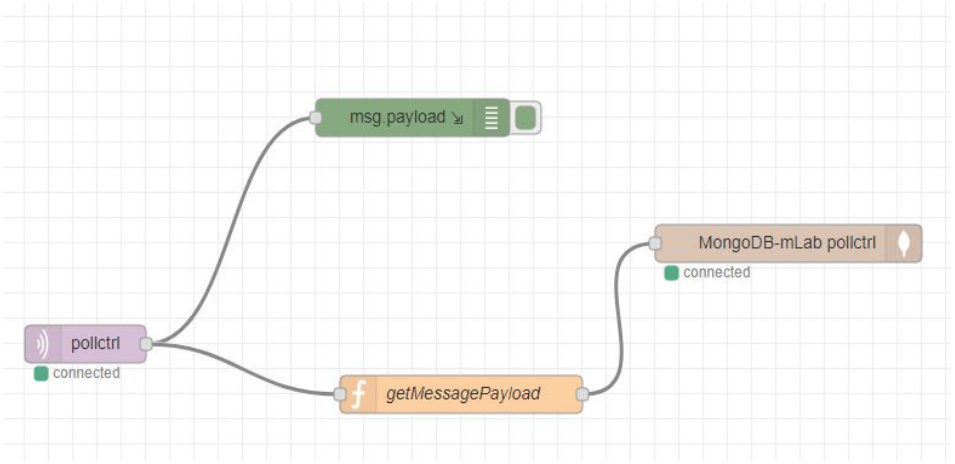

Figura 3.3.3. Flux în Node-RED pe server

În calitate de protocoale de comunicaţii IoT între gateway-urile IoT şi Cloud-ul IoT pentru soluţie, sunt utilizate MQTT (s) şi HTTP (s) - API-REST.

MQTT (Message Queuing Telemetry Transport) este un protocol de comunicare bazat pe conceptul publicare-abonare care trece peste stiva protocoalelor TCP / IP. Filtrarea mesajelor se face de către broker pe teme/subiecte, iar mesajele sunt clasificate pe teme înainte de publicare.

Mecanismul prin care abonaţii primesc doar un subset de toate mesajele publicate se numeşte filtrarea mesajelor. Există două forme de filtrare: filtrare bazată pe subiecte şi filtrare pe conţinut. Abonamentele de publicare sunt gestionate de un broker, care realizeaza filtrarea mesajelor. Toţi clienţii trebuie să stabilească o legătură cu brokerul. Clientul care publică un mesaj de broker este numit editor. Brokerul filtrează mesajele şi le distribuie clienţilor interesaţi pentru a primi un tip specific de mesaj. Clienţii care se abonează la broker pentru a primi mesaje sunt numiţi abonaţi.

Un avantaj al modelului de publicare-abonare este înregistrarea liberă a abonaţilor. Editorii nu trebuie să ştie despre existenţa abonaţilor. Scopul lor principal este de a publica un mesaj către un anumit subiect/canal.

După ce brokerul a fost configurat, clientul MQTT de pe dispozitivele IoT edge poate rula aplicaţiile client MQTT. La pornirea aplicaţiei, un client MQTT este iniţializat pentru a se abona la brokerul găzduit în backend la următoarele subiecte: „/stations” şi „/stations/{MAC_ADDRESS}”, unde MAC_ADDRESS este adresa MAC a cardului de reţea a plăcii de dezvoltare a gateway-ului IoT. Abonarea la aceste subiecte se face pentru a modifica configuraţia staţiei la distanţă. Toate staţiile disponibile pe platformă pot fi modificate publicând o configuraţie în subiectul „/ stations” sau doar o anumită staţie poate fi modificată publicând o configuraţie pe „/stations/adresa” subtopic definit de adresa MAC a plăcii.

Serviciile RESTful sunt implementate prin intermediul platformei NodeJs Adăugarea unui eşantion de date se face printr-o solicitare HTTP POST la “adresa_server/api/probes”.

Cererea POST HTTP trebuie să conţină un pachet în care entitatea este definită în format JSON. Dacă solicitarea nu conţine un corp valid, clientul primeşte un HTTP 400 - o eroare de solicitare greşită.

Componenta front-end a soluţiei IoTP4mSCp este o aplicaţie Single-page (SPA) dezvoltată în Angular şi cu biblioteca de componente Angular Material folosită pentru a utiliza controale vizuale predefinite, precum ferestrele de dialog, butoanele şi tabelele. Proiectul a fost creat folosind interfaţa liniei de comandă (CLI), care facilitează generarea de componente, servicii şi module.

Figura 3.3.4. Componenta Web de monitorizare a nodurilor IoT

Interfaţa Web a utilizatorului include următoarele secţiuni: Zona de utilizator - include autentificarea, crearea contului şi modificarea profilului şi zona staţiei - preia informaţii despre staţiile IoT Gateway, adăugarea / configurarea / ştergerea acestora şi zona de eşantionare a poluării, unde sunt prezentate rapoarte detaliate pe eşantioane înregistrate.

Testarea platformei IoT

Testarea platformei a presupus parcurgerea următorilor paşi

1. Configurarea şi activarea senzorilor IoT;
2. Testarea independentă a componentelor hardware a presupus activarea acestora într-un mediu controlat şi urmărirea comportamentului acestora: dacă reacţionează la modificarea factorilor din mediul extern măsuraţi;
3. Testarea conexiunii dispozitivelor IoT la componentele din backend aflate în platforma cyber range
4. Configurarea şi activarea serviciilor din backend
5. Testarea serverului MQTT prin conectarea la acesta cu o soluţie independentă (https://mqttfx.jensd.de/index.php)
6. Testarea serverului MQTT prin abonarea la anumite subiecte/canale şi publicarea de mesaje
7. Configurarea bazei de date MongoDB prin definirea structurilor în care vor fi stocate datele
8. Configurarea end point-ului REST bazat pe NodeJS
9. Testarea end point-ului REST prin utilizarea unui client independent (https://www.postman.com/)
10. Configurarea fluxurilor Node-RED
11. Testarea conexiunii nodurilor IoT la backend prin abonarea la serverul MQTT şi verificarea logurilor de acces.
12. Testarea conexiunii nodurilor IoT la backend prin apelarea interfeţei REST şi verificarea logurilor de acces.
13. Testarea stocării datelor în baza de date MongoDB
14. Testarea interfeţei Web a aplicaţiei de monitorizare a nodurilor IoT

Activitatea 3.4 - Integrarea şi testarea framework-ului de securitate bazat pe reputaţie pentru IoT

Această activitate a avut ca obiectiv integrarea componentelor şi testarea funcţionalităţilor implementate în cadrul framework-ului de securitate bazat pe reputaţie pentru IoT. ATLAS este un sistem software alcatuit din 3 componente: *client*, *gateway* şi *cloud*, care sunt integrate prin intermediul unei suite de protocoale de comunicaţie. Acest framework oferă un mecanism de gestionare a dispozitivelor IoT din reţea, precum şi un *API* care permite integrarea cu aplicaţii proprietare care rulează la nivelul dispozitivelor IoT. Pe lângă cele 3 componente principale, ATLAS mai conţine o aplicaţie Android care permite autorizarea comenzilor trimise către un dispozitiv IoT client.

ATLAS *Client* constă într-un agent care rulează la nivelul dispozitivelor IoT şi care comunică cu o aplicaţie de *data plane*. Componenta client permite unei aplicaţii proprietare care foloseşte *API*-ul ATLAS să comunice cu gateway-ul. De asemenea, componenta client extrage o serie de informaţii de telemetrie de la nivelul dispozitivului IoT şi le transmite în timp real componentei ATLAS Gateway. La nivelul dispozitivului IoT, împreună cu aplicaţia ATLAS Client, rulează şi aplicaţia ATLAS Data Plane care este un exemplu de integrare a unei aplicaţii proprietare cu sistemul ATLAS.

ATLAS *Gateway* este o componentă software care rulează la nivelul dispozitivului gateway (edge) şi care gestionează nodurile IoT din reţea. Fiecare dispozitiv client (nod IoT) este asociat unui gateway care extrage atât informaţii de sistem (valori de telemetrie), cât şi informaţii ale aplicaţiei proprietare (la nivel de data plane) care rulează pe acelaşi dispozitiv. Componenta ATLAS Gateway rulează la nivelul dispozitivului gateway împreuna cu un broker MQTT (Mosquitto) personalizat care deserveşte comunicaţia dintre aplicaţiile proprietare existente pe dispozitivele de tip client, folosind protocolul MQTT.

Principala componentă a sistemului ATLAS Gateway este modulul de calcul al reputaţiei. Acesta permite crearea unei arhitecturi de tipul S2aaS (Sensing-as-a-Service), în care fiecare dispozitiv client are calculată o valoarea de reputaţie la nivelul gateway-ului pentru o anumită categorie (de exemplu, reputaţie pentru senzorul de temperatură, reputaţie pentru senzorul de presiune etc.). Astfel, un dispozitiv client care consumă date din sistemul ATLAS, colectează date de la alte dispozitive client şi interoghează gateway-ul pentru a afla care dispozitiv care scorul de reputaţie cel mai mare, urmând ca la finalul procesului de consum de date, reputaţia tuturor dispozitivelor care au livrat date să fie actualizată. Modulul gateway sincronizează în timp real toate informaţiile agregate despre clienţi cu componenta ATLAS Cloud. O altă componentă importantă a sistemului ATLAS Gateway este submodulul firewall, care este un plug-in al broker-ului Mosquitto. Submodulul firewall, preia reguli de firewall de la componenta principală ATLAS Gateway şi inspectează fiecare pachet de MQTT redirectat de către brokerul MQTT de la nivelul gateway-ului.

ATLAS *Cloud* este o aplicaţie web care permite administrarea mai multor dispozitive gateway şi client. Aplicaţia Cloud preia date de la Gateway şi le expune către componenta de front-end folosită de administratori. Prin intermediul ei, un utilizator tip *administrator* poate să inspecteze starea sistemul IoT.

ATLAS *Android* este o aplicaţie de smartphone care permite administratorului reţelei IoT locale să aprobe comenzile primite de către un dispozitiv IoT client. Această aplicaţie permite segregarea rolurilor în cadrul sistemului ATLAS, în care avem de pe o parte administratorul reţelei IoT locale şi pe de altă parte administratorul platformei cloud. Astfel, executarea unor comenzi care pot să schimbe starea sistemului IoT trebuie să fie aprobate în timp real de către administratorul reţelei.

Arhitectura generala a sitemului ATLAS este prezentată în Figura 3.4.1. Tot în această figură sunt ilustrare şi protocoalele folosite pentru a asigura comunicaţia între componentele sistemului.

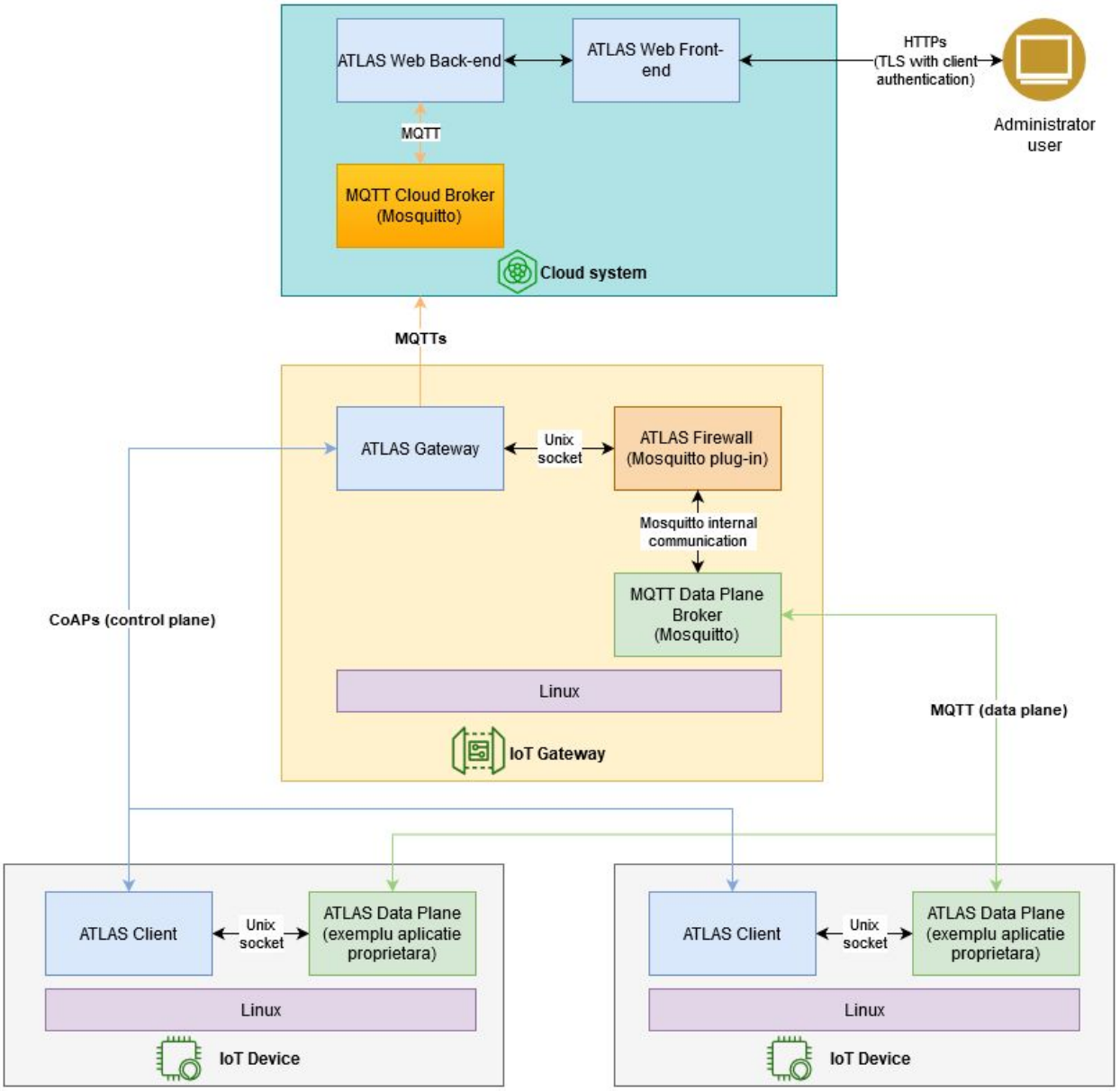

Figura 3.4.1. Arhitectura generală a sistemului ATLAS

Componentele ATLAS Client şi ATLAS Data Plane necesită un sistem de operare Linux. Aplicaţia ATLAS Data Plane este un exemplu de aplicaţie care foloseşte API-ul sistemului ATLAS, aceasta putând fi modificată de către fiecare implementare în parte. Aplicaţiile ATLAS Client şi ATLAS Data Plane comunică prin intermediul unui socket de tip Unix (SCTP), în care ATLAS Client este server, iar ATLAS Data Plane este client. Aplicaţia ATLAS Client comunică la nivel de control plane cu aplicaţia ATLAS Gateway prin intermediul protocolului CoAPs, securizat prin intermediul DTLS-PSK.

La nivelul dispozitivului gateway rulează aplicaţiile ATLAS Gateway şi ATLAS Firewall, cea din urmă fiind un plug-in al brokerului de MQTT Mosquitto. ATLAS Gateway instalează politici la nivelul ATLAS Client şi colectează date în timp real atât de la ATLAS Client cât şi de la ATLAS Data Plane (prin intermediul ATLAS Client). ATLAS Firewall primeşte politici de firewall la nivel aplicaţie (al protocolului MQTT) şi raportează către ATLAS Gateway statistici ale regulilor pentru numărul de pachete acceptate şi rejectate. ATLAS Gateway comunică cu ATLAS Firewall prin intermediul unui socket de tip Unix (TCP), în care ATLAS Gateway este server, iar ATLAS Firewall este client. ATLAS Firewall fiind un plug-in al brokerului Mosquitto, implementează o interfaţă specifică acestei soluţii, fiind compilat sub forma unui *shared object* şi încărcat la runtime de către soluţia Mosquitto. ATLAS Gateway sincronizează în timp real datele clienţilor cu soluţia ATLAS Cloud, prin intermediul protocolului MQTT.

Pentru aceasta, ATLAS Gateway se conectează la un broker MQTT care rulează la nivelul componentei cloud. Conexiunea dintre componenta ATLAS Gateway şi brokerul MQTT cloud este securizată prin intermediul unei conexiuni TLS (certificat de server), iar utilizarea aplicaţiei gateway se realizează prin intermediul unui username şi a unei parole.

ATLAS Gateway este componenta sistemului ATLAS care implementează mecanismul de calcul al reputaţiei pentru dispozitivele client. Algoritmul de calcul al reputaţiei este impactat atât de către componente ATLAS Client: timpul în care un dispozitiv este online, cât şi de către componenta ATLAS Data plane: datele publicate de senzori. Scorul de reputaţie pentru fiecare dispozitiv client este persistent la nivelul gateway-ului şi este sincronizat în timp real cu gateway-ul. Instanţele ATLAS Data plane interoghează ATLAS Gateway pentru a afla identificatorul dispozitivului cu scorul de reputaţie cel mai mare dintr-o anumită categorie (de exemplu, temperatură). Astfel ATLAS Data plane transmite cererea de interogare a scorului de reputaţie instanţei ATLAS Client (comunicaţie socket local), aceasta din urmă comunicând cu gateway-ul prin intermediul protocolului CoAP.

ATLAS Cloud este o aplicaţie Web care preia date de la ATLAS Gateway şi le expune către componenta de front-end a aplicaţiei. Astfel, aplicaţia ATLAS Cloud înglobează un client de MQTT, pe de o parte, şi un server HTTP, pe de altă parte. Aplicaţia ATLAS Cloud este securizată prin intermediul unei conexiuni TLS cu autentificare mutuală, în care administratorul sistemului are un certificat de client. Administratorul sistemului este entitatea care are acces la aplicaţia ATLAS Cloud şi care poate să analizeze metricile tuturor dispozitivelor înrolate în sistem: gateway-uri şi module IoT client.

Sistemul ATLAS permite transmiterea de comenzi de la modulul Cloud la dispozitivul client, comenzi ce pot fi aprobate de către un administrator al reţelei IoT locale prin intermediul unei aplicaţii mobile Android. Administratorul reţelei IoT locale, denumit şi owner, este entitatea care deţine şi instalează reţeaua IoT (formată din dispozitive IoT client şi un gateway), iar administratorul sistemului este entitatea care deţine şi controlează sistemul ATLAS Cloud. Această segregare de roluri conferă flexibilitate în operarea sistemului ATLAS, un administrator al reţelei IoT locale putând să delege monitorizarea sistemului unei alte organizaţii care operează platforma ATLAS Cloud.

Comenzile trimise de către modulul cloud dispozitivelor client, sunt transmise într-o fază iniţială gateway-ului la care este înregistrat clientul, urmând ca acesta să trimită comanda către client. Astfel, un gateway se poate afla în două stări: poate să aibă sau să nu aibă nu un *owner* asociat. În cazul în care gateway-ul nu are un *owner* asociat, comenzile primite de la modulul cloud sunt transmise direct către client. În cazul contrar, la primirea unei comenzi, gateway-ul validează din punct de vedere criptografic (prin verificarea unei semnături) faptul că *owner*-ul a aprobat comanda. Iniţial gateway-ul nu are un owner asociat, acesta permiţând executarea unui proces de *claim* prin intermediul căruia un *owner* instalează o cheie secretă pe gateway. Gateway-ul poate să aibă un singur *owner*, prin urmare poate să execute un singur proces de *claim*.

Planul de testare

Pentru verificarea funcţionalităţilor implementate în cadrul framework-ului de securitate ATLAS a fost elaborat un plan de testare care a avut la bază următoarele principii:

1. Testele vor avea un anumit grad de flexibilitate, putând fi adaptate, dacă situaţia impune acest lucru.
2. Testele vor fi parcurse folosind aceleaşi mecanisme de toţi membrii echipei de testare.
3. Testele care vor avea calificativul parţial “FAILED” vor fi refăcute pornind de la cea mai apropiată stare anterioară (în cazul în care testul nu necesită resetarea întregului sistem).
4. Testele vor putea fi refăcute de oricâte ori, necesitând doar aducerea sistemului la momentul anterior iniţierii respectivului test.

5. Testele vor fi împărţite în paşi succesivi, pentru a facilita reluarea unei activităţi, precum şi pentru a reduce riscul de identificare eronată a momentului apariţiei unei erori în sistem.

Funcţie de obiectivul urmărit, testele au fost împărţite pe categorii, astfel:

Testarea integrării componentelor principale. Testarea de integrare a componentelor a presupus evaluarea interacţiunii dintre componente. Vizualizarea interacţiunilor s-a realizat individual prin testare manuală a următoarelor activităţi:

- Pornirea portalului
- Înregistrarea unui client la nivel de gateway
- Înregistrarea unui gateway la nivel de portal
- Pornirea gateway-ului
- Pornirea clientului
- Pornirea brokerului de MQTT care rulează la nivelul gateway-ului
- Pornirea data plane-ului şi verificarea finalizării cu succes a acestora

Testarea funcţionalităţilor de securitate. Testarea funcţionalităţilor de securitate a presupus evaluarea modului de supervizare de către framework-ul ATLAS a sistemului IoT peste care este instalat. Vizualizarea stării sistemului s-a realizat prin intermediul portalului Web disponibil la nivelul componentei ATLAS Cloud. Principalele activităţi urmărite au fost:

- Verificarea instalării alarmelor pentru datele de telemetrie şi extragerea acestor date;
- Verificarea instalării politicilor de firewall la nivel gateway şi eficienţa filtrării pachetelor din sistem;
- Verificarea modului de preluare, de către componenta client, a informaţiilor de la aplicaţia proprietară care rulează pe nodul IoT (simulată prin aplicaţia *data_plane*);
- Verificarea persistenţei setărilor de securitate în cazul pierderii conexiunii sau întreruperii alimentării.

Testarea modulului de calcul al reputaţiei clienţilor. Testarea modulului de calcul al reputaţiei dispozitivelor IoT conectate la un gateway (şi pe care rulează componenta ATLAS Client) a avut rolul de a evidenţia modul în care sistemul răspunde atât în cazul unui trafic normal, cât şi în cazul apariţiei unor actori maliţioşi care încerca destabilizarea sistemului (fie prin injectarea unor date eronate, fie prin atacuri fizice asupra nodurilor IoT). Reprezentând un modul critic al componentei ATLAS Gateway, testarea acestuia a fost iniţiată din etapa dezvoltării componentelor ATLAS, prin implementarea unui modul software complementar care preia funcţionalităţile de calcul a reputaţiei şi stabileşte diferite scenarii de test care oferă o serie de parametri ce pot fi personalizaţi (modulul ***atlas_gateway_reputation_test***). Suplimentar acestui modul, în cadrul testelor au fost simulate următoarele scenarii de utilizare a sistemului IoT:

- Verificarea dependenţei reputaţiei de sistem (control plane) de stabilitatea conexiunii şi performanţa nodului;
- Verificarea dependenţei reputaţiei datelor (data plane) de feedback-ului dat de consumatori producătorilor;
- Verificarea modificării reputaţiei în cazul unui atac.

Testarea de performanţă sub stres. Aceste teste au vizat modul de funcţionare a framework-ului în condiţii mai puţin “favorabile”, cum ar fi:

- Pornirea simultană a mai multor procese la nivelul clientului;
- Memorie RAM aproape ocupată;
- Simularea unui trafic MQTT care să depăşească valorile de policy stabilite;
- Diferite tipuri de întreruperi (de exemplu, oprirea şi repornirea unei componente).

Testarea modulului de autorizare comenzi. Acest tip de testare a verificat capabilitatea framework-ului de securitate de a permite unui utilizator de tip owner să execute o acțiune de claim asupra dispozitivului gateway astfel încât toate comenzile care sunt trimise de către platforma cloud către dispozitivul client (prin intermediul gateway-ului) să fie aprobate de către acesta prin intermediul aplicației Android.

Mediul de test

Pentru asigurarea unui mediu de test cât mai apropiat de un sistem real, platforma de test a fost compusă din elemente hardware și software eterogene. Fiecare componentă a framework-ului de securitate a fost configurată și rulată pe câte un dispozitiv independent, după cum urmează:

- modulul ***atlas_client*** a fost rulat de pe dispozitive Raspberry Pi 3 model B+ (Raspbian GNU/Linux 10 (buster));
- modulul ***atlas_gateway*** a fost rulat de pe dispozitive Odroid XU4Q (Ubuntu 20.04.1 LTS (Focal Fossa)) și a inclus o variantă personalizată a broker-ului MQTT, Eclipse Mosquitto;
- modulul ***atlas_cloud*** a fost rulat de pe o mașină virtuală care simula o arhitectură hardware de tip server, având instalat ca sistem de operare: Ubuntu 20;
- **aplicația Android** a fost rulată pe un smartphone cu sistem de operare Android 9.

Suplimentar, au mai fost utilizate 4 mașini virtuale standard de tip PC (având instalat Ubuntu 20) pentru simularea unor componente de tip Gateway și Client. Mediul de test folosit este descris în Figura 3.2.4.

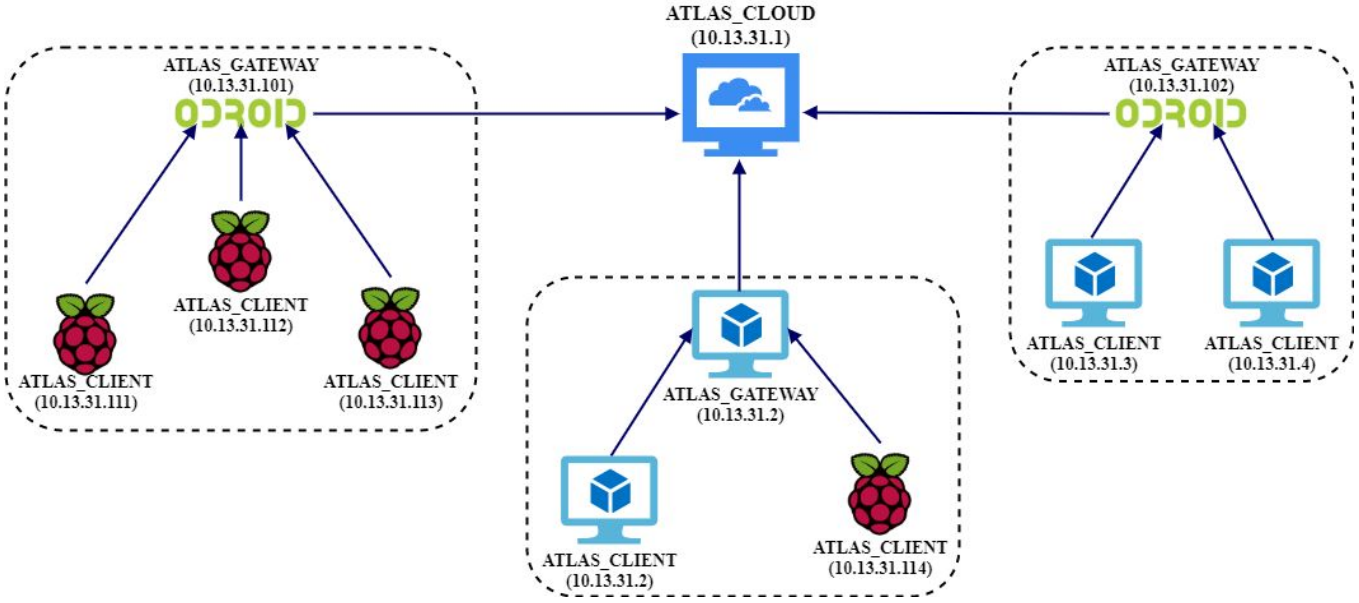

Figura 3.4.2. Mediul de test

Rezultatele testării

Pentru fiecare test s-au definit scopul, condițiile de testare, acțiunile, rezultatul așteptat, rezultatul obținut și concluzia finală. În total, au fost efectuate **48 de teste**, rezultatul fiind pozitiv pentru fiecare test în parte. Lista completă a testelor efectuate este următoarea:

1. Creare elemente identificare client
2. Creare elemente identificare gateway
3. Înregistrare client la nivel de gateway
4. Înregistrare gateway la nivel de portal
5. Ștergere gateway
6. Validare la nivel de portal a înregistrării unui client
7. Înregistrare data plane

8. Fail-back data_plane (scenariu 1)
9. Fail-back data_plane (scenariu 2)
10. Fail-back client
11. Validare unicitate identitate gateway la nivelul portalului
12. Verificare funcţionare sistem ATLAS în cazul schimbării adresei IP a clientului
13. Failback gateway
14. Trimitere statistici PacketsPerMinute şi PacketsAvgLength
15. Verificare politică de firewall “Egress passed packets”
16. Verificare politici de firewall “Egress dropped packets” şi “Ingress dropped packets”
17. Verificare politici de firewall “Egress passed packets” şi “Ingress passed packets”
18. Verificare politici de firewall “Egress dropped packets” şi “Ingress dropped packets”
19. Validare persistenţă reguli de firewall
20. Validare actualizare reguli de firewall după repornirea gateway-ului
21. Verificare valoare telemetrie Hostname
22. Verificare valoare telemetrie SysinfoProcs
23. Verificare valoare telemetrie SysinfoUptime
24. Verificare valoare telemetrie KernelInfo
25. Verificare valori telemetrie SysinfoFreeram, SysinfoTotalram şi SysinfoSahredram
26. Verificare valori telemetrie SysinfoFreeswap şi SysinfoTotalswap
27. Verificare valoare telemetrie SysinfoBufferram
28. Verificare valori telemetrie de încărcare a sistemului la 1/5/15 minute
29. Verificare valori telemetrie pentru IP şi port client
30. Validare dependenţă reputaţie de sistem (control plane) de timpul în care un client este înregistrat la gateway
31. Validare dependenţă reputaţie de sistem (control plane) de numărul de pachete trimise şi acceptate de către alte dispozitive
32. Validare reputaţie de data plane
33. Validare dependenţei reputaţiei de data plane faţă de corectitudinea datelor furnizate de un dispozitiv
34. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul nu are asociat un owner
35. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul nu are asociat un owner (failover gateway)
36. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul nu are asociat un owner (failover client)
37. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul primeşte mai multe comenzi la rând
38. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul primeşte mai multe comenzi la rând (failover gateway)
39. Validarea procesului de asociere al unui gateway cu un owner (proces de claim folosind aplicaţia mobilă)
40. Validarea faptului că un gateway care are asociat un owner nu poate să fie asociat altui owner
41. Validarea faptului că toate comenzile pentru un gateway care are asociat un owner trebuie să fie aprobate de către owner prin intermediul aplicaţiei mobile
42. Validarea faptului că toate comenzile pentru un gateway care are asociat un owner trebuie să fie aprobate de către owner prin intermediul aplicaţiei mobile (failover gateway)
43. Validarea faptului că toate comenzile pentru un gateway care are asociat un owner trebuie să fie aprobate de către owner prin intermediul aplicaţiei mobile (failover client)

44. Validarea faptului că toate comenzile trimise către un client sunt vizibile în aplicaţia mobilă chiar dacă aceasta este închisă
45. Validarea faptului că un gateway se poate şterge de la nivelul aplicaţiei mobile
46. Validarea faptului că modulul cloud suportă mai multe dispozitive gateway asociate cu owneri diferiţi
47. Validarea faptului că modulul cloud suportă mai multe dispozitive gateway asociate cu owneri diferiţi (failover două gateway-uri)
48. Validarea comenzii de RESTART şi SHUTDOWN în scenariul în care gateway-ul primeşte comenzi pentru doi clienţi diferiţi

În concluzie, apreciem că soluţia dezvoltată satisface cerinţele de proiectare şi asigură un nivel de performanţă ridicat. Soluţia se bazează pe tehnologii moderne, are o arhitectură modulară, putând fi uşor extinsă prin adăugarea de noi funcţionalităţi. Framework-ul de securitate realizat poate fi uşor integrat în scenarii de tip smart home sau smart city precum şi în aplicaţii de tip mobile crowd sensing.

Proiect Component 3 - Mediu distribuit pentru exerciţii şi cercetare avansată în domeniul securităţii cibernetice

Activitatea 3.5 - Interconectarea platformelor cyber range şi elaborarea de scenarii pentru exerciţii de apărare cibernetică

Această activitate a avut ca obiective replicarea platformei cyber range la nivelul tuturor universităţilor partenere şi constituirea de repository-uri cu scenarii de exerciţii de apărare cibernetică. Scopul final al acestui proiect component îl reprezintă federarea platformelor cyber range, fapt ce va permite partajarea de cursuri şi laboratoare, organizarea în comun de exerciţii şi dezvoltarea şi testarea de noi soluţii de securitate cibernetică.

Principalele componentele ale platformei cyber range sunt prezentate în Figura 3.5.1. **Infrastructura hardware** este formată din servere, storage şi echipamente de reţea (switch-uri, routere, firewall, etc) necesare pentru a crea mediul de lucru pentru instruire / antrenament / experimentare. Infrastructura hardware nu este omogenă la nivelul universităţilor partenere ci diferă funcţie de resursele disponibile şi cerinţele specifice.

Tehnologia de virtualizare folosită în cadrul platformei cyber range este VMware. Mai exact, se foloseşte *VMware vSphere Essentials Plus Kit* care oferă cadrul necesar separării arhitecturii logice de configuraţia hardware şi *vCenter* care oferă posibilităţi avansate de administrare a infrastructurii virtuale şi metode de REST API pentru interfaţarea cu sistemele de automatizare. Componenta vCenter este comandată de către orchestrator cu ajutorul scripturilor Ansible şi asigură controlul tuturor maşinilor şi subreţelelor virtuale definite prin intermediul switch-ului virtual şi VMWare Tools.

Uneltele pentru automatizare şi managementul configuraţiilor au rolul de a automatiza procesul de instanţiere şi replicare a maşinilor virtuale şi reţelelor necesare pentru derularea unui scenariu de instruire sau antrenament. În acest scop se folosesc scripturi (playbooks) Ansible care descriu acţiunile ce trebuie executate pentru configurarea unui sistem sau pentru automatizarea operaţiunilor de provizionare sau deprovizionare a sistemului respectiv. Interacţiunea cu mediul de virtualizare se realizează prin intermediul vCenter.

Infrastructura ţintă reprezintă mediul creat pentru derularea exerciţiilor sau a experimentelor. Funcţie de scenariu, infrastructura ţintă poate emula un singur calculator sau o reţea organizaţională complexă. Mediul astfel creat este izolat de reţelele de producţie, din motive de securitate. Conectarea utilizatorilor

la mediul de lucru se face prin OpenVPN iar accesul la sistemele şi echipamentele emulate se face prin conexiuni SSH sau Remote Desktop.

Sistemul de orchestrare este liantul care uneşte toate componentele astfel încât să poată fi implementat scenariul proiectat. Sistemul de orchestrare este compus dintr-o serie de scripturi Python care instanţiază infrastructura ţintă plecând de la un pool de maşini virtuale cu imagini de bază ale sistemelor de operare, echipamentelor de reţea şi tehnologiilor de securitate larg răspândite.

Accesul la platforma cyber range se face prin intermediul unui **Portal Web**, dezvoltat în PHP**.** Acesta reprezintă interfaţa utilizator (front end) şi este utilizat de către:

- administratorii pentru modificarea unor servicii esenţiale sau reconfigurarea conexiunilor dintre cyber range şi echipamentele fizice;
- instructori pentru definirea de template-uri de maşini virtuale, definirea şi configurarea de scenarii, alocarea de resurse, lansarea în execuţie a laboratoarelor / exerciţiilor, monitorizarea resurselor alocate, dealocarea resurselor la finalul activităţilor;
- studenţi pentru obţinerea profilului OpenVPN pentru conectarea în infrastructură şi accesarea maşinilor virtuale alocate.

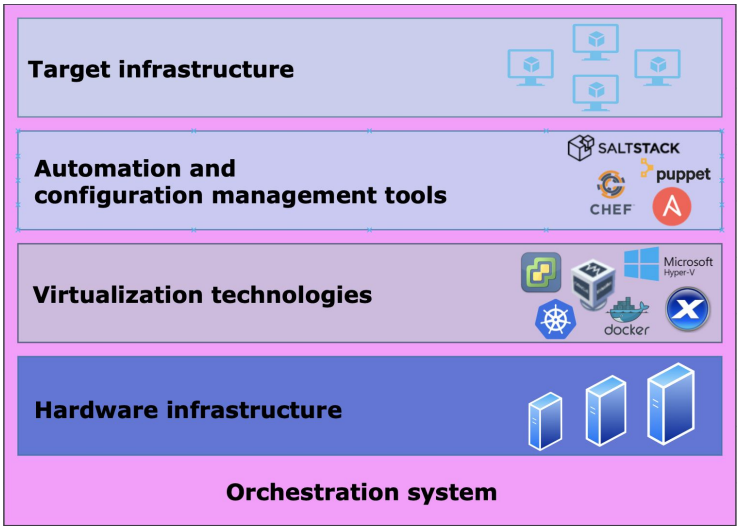

Figura 3.5.1. Componentele platformei cyber range

În plus faţă de componentele de bază, platforma cyber range poate fi augmentată cu sisteme de tip LMS (Learning Management System) sau sisteme pentru emularea activităţii utilizatorilor, traficului de reţea şi serviciilor Internet. Toate acestea permit crearea unui mediu de antrenament cât mai apropiat de realitate, atât din punct de vedere operaţional cât şi funcţional.

În continuare este prezentat stadiul implementării platformei cyber range la nivelul fiecărei universităţi, modul în carea aceasta a fost testată şi scenariile de exerciţii de apărare cibernetică elaborate.

Academia Tehnică Militară

La nivelul Academiei Tehnice Militare (ATM), platforma cyber range a fost instalată şi testată în cadrul Centrului de Excelenţă pentru Tehnologii Avansate de Securitate Cibernetică. Infrastructura hardware folosită pentru instalare a fost formată din echipamente existente în cadrul centrului sau achiziţionate în cadrul proiectului. Echipamentele sunt de ultimă generaţie, utilizate pe larg în mediul academic şi în industria de profil:

- 3 servere CISCO UCS M4, dual Xeon, 2 x 8 core (2x16 thread), 128GB RAM, rezultând un disponibil de 48 core-uri fizice / 96 thread-uri şi 384GB RAM pentru lansarea în execuţie a infrastructurii necesară derulării unui exerciţiu;
- 1 storage hibrid EMC VNXe3200, pentru stocarea template-urilor şi maşinilor virtuale folosite pe parcursul rulării unui exerciţiu. Echipamentul este proiectat să asigure viteze sporite de acces, prin utilizarea a 2 SSD-uri ca buffere de citire-scriere şi HDD-uri de viteză mare (10.000 rpm). În momentul de faţă, capacitatea de stocare totală este de aproximativ 6TB;
- 2 switchuri Cisco SG550X-8F8T 16-port 10G pentru asigurarea redundanţei şi a unei conexiuni de 2 x 10Gbps între servere, echipamentul de stocare şi celelalte componente ale infrastructurii;
- 1 firewall Fortinet FortiGate 300E pentru asigurarea securităţii infrastructurii.

După o testare pe componente funcţionale şi servicii dezvoltate în diferite scenarii de utilizare, platforma cyber range a fost operaţionalizată şi folosită pentru derularea de laboratoare la următoarele cursuri de securitate cibernetică din cadrul programelor de licenţă / master / specializare:

- Cursul *Testarea Securităţii Sistemelor Informatice* din cadrul programului de licentă *Ingineria şi Securitatea Sistemelor Informatice Militare*;
- Cursurile *Protocoale de Securitate*, *Managementul Securităţii Informaţiilor, Război Cibernetic* din cadrul programului de master *Securitatea Tehnologiei Informaţiei*;
- Cursul *Securizarea Sistemelor de Operare şi a Serverelor de Aplicaţie* din cadrul programului de specializare *Tehnologii de Apărare Cibernetică*.

Testele iniţiale au relevat importanţa majoră a vitezei de acces la mediile de stocare pentru maşinile virtuale şi au condus la integrarea în cadrul platformei a unor echipamente de stocare cu acces mult mai rapid (all-flash). Testarea integrată a serviciilor platformei cu un număr semnificativ de utilizatori (studenţi şi cursanţi) a permis verificarea gradului de flexibilitate şi scalabilitate al soluţie. Pentru a putea fi folosită în scenarii cât mai diverse, au fost dezvoltate funcţionalităţi noi cum ar fi modulul de tip firewall pentru interconectarea / izolarea utilizatorilor şi resurselor.

Pe baza concluziilor rezultate în urma testelor derulate în ATM, au fost elaborate specificaţiile şi s-a realizat achiziţia celui de-al treilea lot de echipamente necesare pentru replicarea platformei cyber range la nivelul celorlalte universităţi.

Pe parcursul derulării proiectului, în cadrul ATM au fost elaborate 3 scenarii pentru exerciţii de apărare cibernetică, astfel:

Scenariul 1 ATM - Source code leak

În mediul online au fost publicate părţi din codul sursă şi referinţe la componente din cadrul unei soluţii software de securitate dezvoltată de o companie. În aceeaşi zi, unul din dezvoltatorii software ai companiei a raportat faptul că nu reuşeşte să acceseze server-ul de dezvoltare pe care sunt stocate module din cadrul proiectului. Participanţii la exerciţiu fac parte dintr-o echipă de răspuns la incidente de securitate cibernetică care trebuie să investigheze incidentul - să recapete controlul asupra server-ului şi fişierelor sursă, şi să descopere dacă acest incident a fost folosit de către atacatori ca punct de pivotare pentru atacuri în reţeaua locală a companiei. Acest scenariu a fost dezvoltat şi utilizat în cadrul exerciţiului naţional de securitate cibernetică CyDEx 2019.

Fiecare participant la exerciţiu primeşte acces prin VPN la un POD de resurse în care sunt disponibile 3 maşini virtuale Linux - serverul de dezvoltare (Ubuntu server, 1 vCPU, 2GB RAM), staţia utilizată pentru recâştigarea accesului la server (Kali Linux, 1 vCPU, 2GB RAM) şi appliance-ul virtual SIFT (1 vCPU, 2GB RAM) pentru investigarea mecanismelor de infectare şi persistenţă (vezi Figura 3.5.2).

Obiectivele acestui scenariu sunt:

1. Exersarea tehnicilor utilizate în red teaming pentru detecţia şi exploatarea vulnerabilităţilor, precum şi escaladarea privilegiilor în vederea obţinerii accesului la resurse
2. Exersarea capabilităţilor de investigare a sistemelor Linux şi identificarea vectorului de infecţie

Etapele pentru finalizarea cu succes a scenariului:

1. Utilizarea tehnicilor de Red-Teaming pentru identificarea şi exploatarea unor vulnerabilităţi la nivelul serverului de dezvoltare
2. Recuperarea accesului cu drepturi de administrare la nivelul serverului
3. Investigarea artefactelor lăsate de către atacator pe server (folosit ulterior în scop ilicit, pe baza unor fişiere maliţioase)

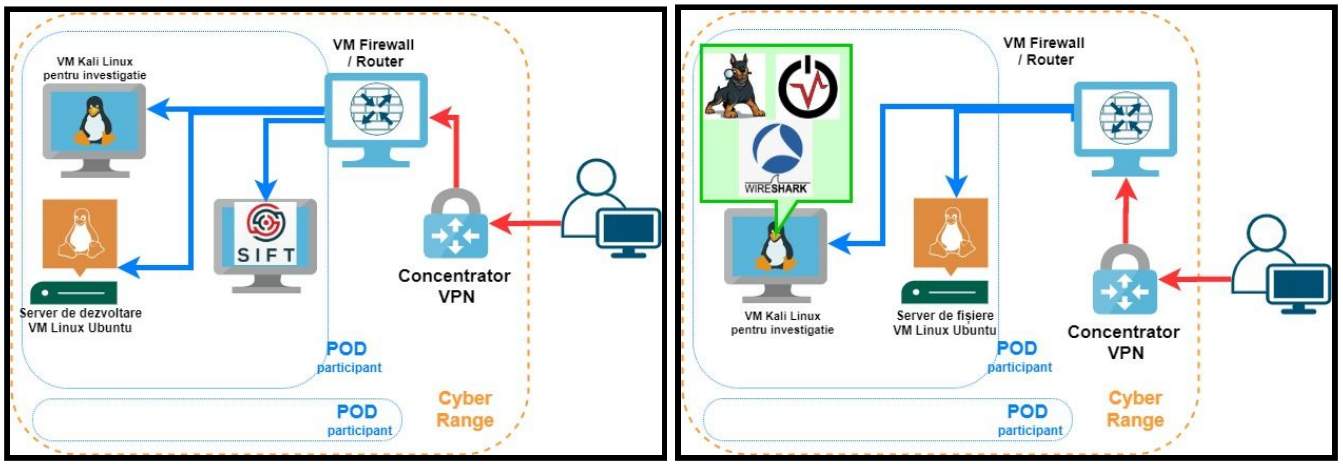

Figura 3.5.2. Infrastructura pentru scenariile 1 şi 2 ATM

Scenariul 2 ATM - Telemuncă în pandemie

În contextul pandemiei de Covid-19, angajaţii unei instituţii lucrează de la distanţă, utilizând şi o platformă de transfer a documentelor prin care angajaţii îşi pot încărca arhive. Aplicaţia în versiunea beta a fost expusă urgent în Internet pe un server Ubuntu. În urma unui atac, serverul a fost compromis (acces restricţionat) şi s-a constatat un volum important de date exfiltrate în mod criptat. Acest scenariu a fost dezvoltat şi utilizat în cadrul exerciţiului naţional de securitate cibernetică CyDEx 2020.

Fiecare participant la exerciţiu primeşte acces prin VPN la un POD de resurse în care sunt disponibile 2 maşini virtuale Linux - serverul de fişiere (Ubuntu server, 1 vCPU, 2GB RAM) şi staţia utilizată pentru investigarea incidentului (Kali Linux, 2 vCPU, 2GB RAM) (vezi Figura 3.5.2).

Obiectivele acestui scenariu sunt:

1. Exersarea tehnicilor de analiză cod şi detectare vulnerabilităţi
2. Exersarea tehnicilor de digital forensics pentru depistarea vectorului de infecţie
3. Dezvoltarea unor capabilităţi de analiză, prelucrare şi decriptare trafic

Etapele pentru finalizarea cu succes a scenariului:

1. Determinarea vulnerabilităţii serverului - analiza codului sursă a server-ului şi identificarea secţiunilor care îl fac vulnerabil
2. Analiza arhivei maliţioase - analiza arhivei care, odată încărcată pe platformă, a produs infectarea server-ului
3. Identificarea datelor exfiltrate - construirea unui mecanism de decriptare a unor date exfiltrate, dintr-o captură de trafic
4. Oprirea exfiltrării - identificarea condiţiilor de oprire a exfiltrării datelor

Scenariul 3 ATM - Pentest aplicaţie Web

În urma unor investigaţii este identificată o subreţea în care este suspectată existenţa unei aplicaţii Web vulnerabilă. Fiecare participant la exerciţiu primeşte acces prin VPN la un POD de resurse în care dispune de o maşină virtuală preconfigurată cu tool-urile necesare (Kali Linux, 2 vCPU, 2GB RAM). Fiecare POD al unui participant este configurat cu un VLAN propriu dar care să permită accesarea serverului ţintă, comun pentru toţi participanţii (Ubuntu server, 8 vCPU, 4GB RAM). (vezi Figura 3.5.3)
Opţional, în funcţie de timpul alocat rezolvării scenariului precum şi nivelul de dificultate stabilit, în VLAN-ul serverului ţintă poate fi conectat generatorul de trafic IXIA Breaking Point care să ruleze în modul one-arm (o singură interfaţă fizică de reţea necesară) sau two-arm (2 interfeţe fizice care să asigure conexiuni client-server). Astfel, se pot simula o multitudine de host-uri sau servicii active.

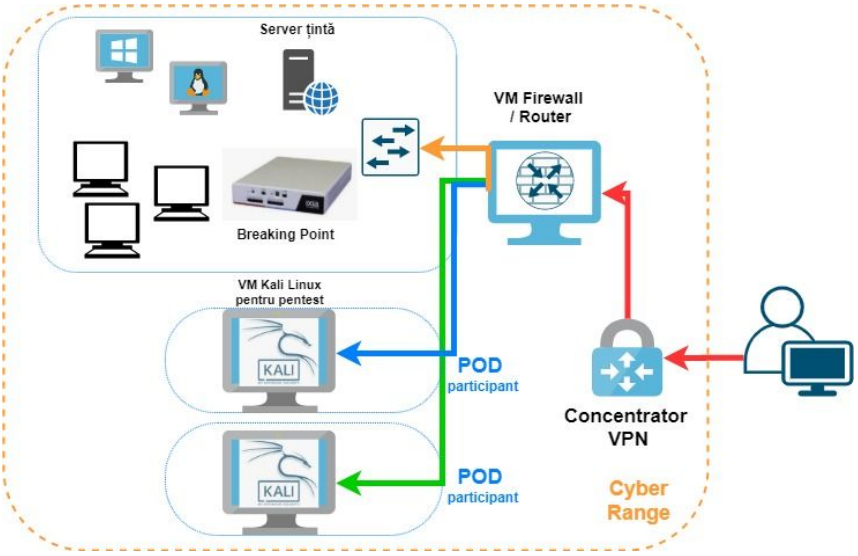

Figura 3.5.3. Infrastructura pentru scenariul 3 ATM

Obiectivele acestui scenariu sunt:

1. Exersarea tehnicilor scanare a reţelei
2. Optimizarea tehnicilor de scanare
3. Exploatarea unor vulnerabilităţi cunoscute în cadrul unui server Web

Etapele pentru finalizarea cu succes a scenariului:

1. Scanarea reţelei pentru detectarea IP-urilor active
2. Optimizarea scanării pentru obţinerea serviciilor active şi identificarea unui serviciu potenţial vulnerabil
3. Decodificarea unor fişiere de configurare
4. Execuţia unui atac de tip brute force pentru obţinerea de fişiere şi directoare
5. Exploatarea unor scripturi PHP
6. Extragerea unor date din imagini (steganografie)

Universitatea Politehnica din Bucureşti

În cadrul Universităţii Politehnica din Bucureşti (UPB) studenţii folosesc multiple resurse digitale printre care dorim doar sa amintim de platforma Moodle (https://curs.upb.ro/) si platforma de administrare a identitatii digitale (https://my.upb.ro/login). Menţionăm ca fiecare angajat UPB deţine o identitate digitală şi că instituţia oferă o metoda de autentificare unitară prin standardul Security Assertion Markup Language.

Infrastructură hardware achizitionata în anul 2020 prin intermediul proiectului ATLAS a fost instalată separat de resursele universităţii în datacenterul acesteia, fiind folosită sub forma unui poligon de securitate cibernetica. Configuraţia hardware a echipamentelor este următoarea:

- 2 servere Cisco UCS C220 M5, fiecare avand 2 x 1.9TB 2.5 inch Enterprise Value 6G SATA SSD, 1 x 240GB SATA M.2, 2 x Intel 4216 2.1GHz 22MB Cache (https://ark.intel.com/content/www/us/en/ark/products/193394/intel-xeon-silver-4216-processor-22m-cache-2-10-ghz.html) cu 16 core (32 threads), 512 GB RAM; rezultand un disponibil de 64 de core-uri fizice / 128 thread-uri, 1TB RAM şi 8 TB spaţiu de stocare.

Accesul la acest cyber range se realizeaza printr-o conexiune de tip virtual private network (VPN) cu un echipament hardware PaloAlto 5050, cu un uplink de 1 Gbps. De asemenea mentionam pentru uniformizarea platformelor cyber range software-ul instalat pe echipamente este VMWare ESXi 7.0 cu administrarea prin VMWare VSphere 7.0.

Instalarea, configurarea şi testarea infrastructurii prezentate a fost realizata de către membrii proiectului ATLAS. Platforma operaţională este folosită cu succes la următoarele cursuri de securitate cibernetică din cadrul programelor de master sau licenta:

- Cursul de “Testarea Securităţii Sistemelor de Calcul” din cadrul programului de licenta “Calculatoare şi tehnologia informaţiei”
- Cursul de “Securizarea Reţelelor cu Echipamente Dedicate” din cadrul programului de master “Securitatea Reţelelor Informatice Complexe”
- Cursul de “Cyberdefense şi cyberintelligence” din cadrul programului de master “Sisteme Avansate de Securitate” (în limba engleza).

Platforma de cyber range poate fi accesata de catre orice student/cadru didactic care deţine un cont la UPB. Echipamentul de securitate PaloAlto a fost configurat astfel incat sa permita identificarea la VPN prin standardul SAML şi folosind furnizorul de identitate oferit de către UPB.

Odată autentificati la serviciul de VPN, studenţii pot accesa diversele scenarii puse la dispozitie prin intermediul platformei de cyber range. Menţionăm că platforma a fost testată cu peste 200 de studenţi (la master sau la licenta), validand astfel configurarea, flexibilitatea şi scalabilitatea soluţiei. Mai jos vom prezenta cele 3 scenarii dezvoltate în cadrul proiectului de către UPB.

Scenariul 1 UPB - Descoperirea serviciilor de reţea

Acest scenariu este alcatuit din 2 maşini virtuale pentru fiecare student (vezi Figura 3.5.4). Prima masina virtuala este configurată astfel incat sa emuleze o retea virtuala formată din mai multe servicii şi echipamente de reţea. În acest scenariul studentul are acces doar la masina virtuala care rulează Kali Linux. Maşinile virtuale trebuiesc configurate cu minim 1vCPU şi 1GB memorie RAM.

Obiectivele acestui scenariu sunt:

1. Folosirea utilitarului Wireshark pentru disectia protocoalelor de reţea.
2. Identificarea automată a tuturor dispozitivelor de reţea.
3. Identificarea automată a serviciilor care rulează pe un server.
4. Folosirea protocolului DNS pentru descoperirea numelor folosite în reţea.

Etapele pentru finalizarea cu succes a scenariului:

1. Activarea protocolului DHCP pentru conexiunea cu reţeaua ce trebuie descoperită.
2. Disecarea protocolului DHCP pentru a analiza opţiunile primite precum adresa IP, rutele statice, etc.
3. Folosirea optimă a utilitarului nmap pentru a identifica toate staţiile existe într-un spaţiul de adrese foarte mare 10.0.0.0/8.
4. Folosirea utilitarului nmap pentru descoperirea serviciilor care rulează pe fiecare staţie din reţea.
5. Identificarea configuraţiilor existe pe serverul de DNS din reţea.

6. Identificarea şi accesarea serverului de web din reţea.

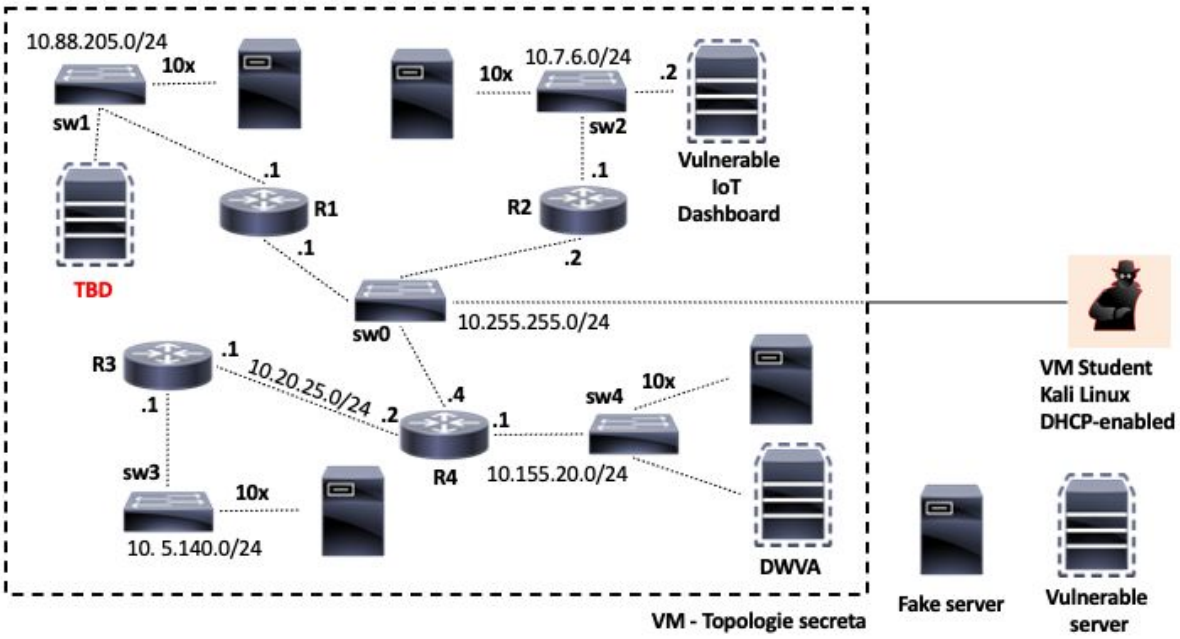

Figura 3.5.4. Scenariul 1 UPB - Topologia secreta

Scenariul 2 UPB - Protejarea împotriva unui atac de tip MITM

Acest scenariul este compus din 3 maşinii virtuale interconectate prin intermediul unui switch virtual (vezi Figura 3.5.5). Toate maşinile virtuale folosesc un sistem de operare Linux, 2 masini vor juca rolul de victimă şi furnizarea accesului la Internet, iar cea de-a treia are rolul de a realiza un atac de tip Man-In-The-Middle (MITM). Maşinile virtuale trebuiesc configurate cu minim 1vCPU şi 1GB memorie RAM.

Obiectivele acestui scenariu sunt:

1. Înţelegerea protocolului ARP
2. Crearea unui atac de tip MITM la nivelul 2
3. Folosirea programului “ettercap”
4. Inspectarea şi modificarea pachetelor de reţea

Etapele pentru finalizarea cu succes a scenariului:

1. Înţelegerea reţelei în care atacatorul se afla prin inspectarea pachetelor de retea
2. Realizarea unui atac de tip ARP poisoning între victima si gateway
3. Capturarea pachetelor generate de către victimă
4. Inspectia pachetelor şi identificarea protocoalelor folosite (DNS şi HTTPS)
5. Alterarea pachetelor de tip ICMP generate de către gateway
6. Modificarea răspunsurilor primite de către victimă la cererile de DNS efectuate
7. Vizualizarea traficului criptat de HTTPS

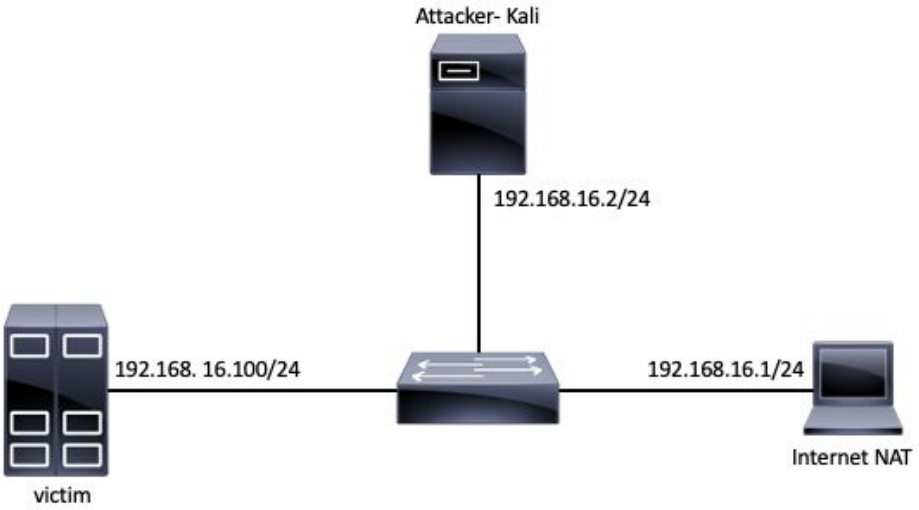

Figura 3.5.5. Scenariul 2 UPB - Identificarea unui atac MITM

Scenariul 3 UPB - Exfiltrarea datelor

Acest scenariu îşi propune să arate multiplele mecanisme prin care un atacator poate exfiltra date din reţea. Cele mai multe atacuri din lumea reala necesita crearea unui canal de comunicare intre masina victimă şi atacator, iar acest scenariu vine în ajutorul studenţilor pentru a identifica traficul de reţea folosit de către atacatori.

Pentru acest scenariu sunt necesare 4 maşinii virtuale, H1 si H2 vor fi folosite pentru crearea canalelor de comunicaţie, o maşină va fi folosit pentru a oferi accesul la Internet, iar maşina IDS va avea scop de vizualizare a traficului de reţea (vezi Figura 3.5.6). Maşinile virtuale trebuiesc configurate cu minim 1vCPU şi 1GB memorie RAM.

Obiectivele acestui scenariu sunt:

1. Identificarea metodelor prin care se pot exfiltra date
2. Identificarea metodelor prin care se pot realiza canale de comunicare de către atacator
3. Analiza traficului de reţea
4. Crearea de canale de comunicare folosind multiple protocoale

Etapele pentru finalizarea cu succes a scenariului:

1. Crearea unui canal de comunicare bazat pe TCP folosind “nc”
2. Crearea unui canal de comunicare bazat pe UDP folosind “nc”
3. Crearea unui canal de comunicare bazat pe HTTP folosind suita “httptunnel (hts & htc)”
4. Crearea unui canal de comunicare bazat pe SSH
5. Crearea unui canal de comunicare bazat pe ICMP folosind “ptunnel”
6. Crearea unui canal de comunicare bazat pe DNS folosind “dns2tcp”
7. Salvarea, inspectarea şi discutarea pachetelor de reţea pentru fiecare canal de comunicare realizat.

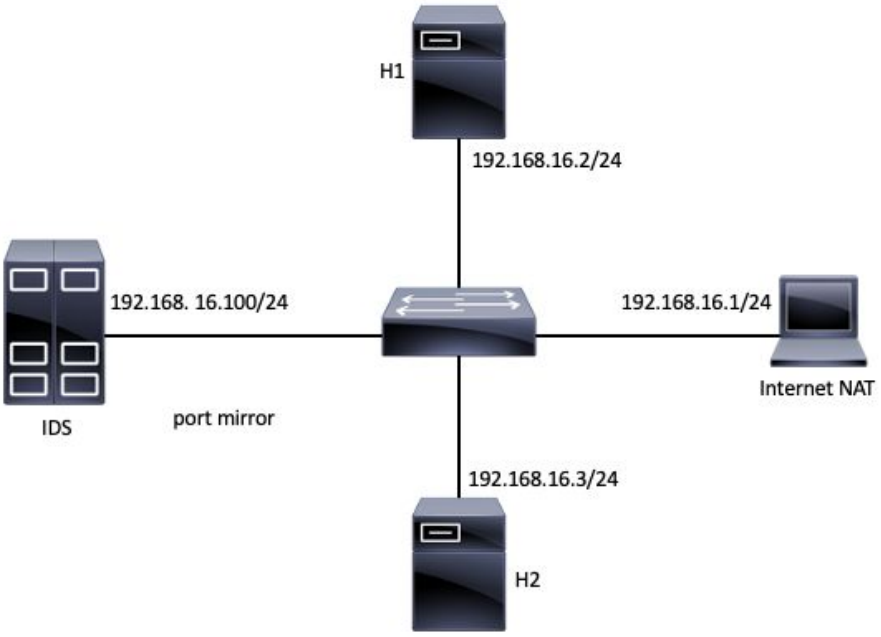

Figura 3.5.6. Scenariul 3 UPB - Exfiltrarea datelor

Academia de Studii Economice din Bucureşti

La Academia de Studii Economice din Bucureşti (ASE), platforma cyber range a fost instalată în cadrul Data Center-ului administrat de Direcţia IT a universităţii şi a fost testată în cadrul activităţilor specifice proiectului, dar şi în cadrul unor cursuri aferente programului de masterat de Securitatea Informatică. Infrastructura hardware folosită pentru instalare a fost formată din echipamente existente în cadrul centrului sau achiziţionate în cadrul proiectului. Echipamentele sunt de ultimă generaţie, utilizate pe larg în mediul academic şi în industria de profil:

- 2 servere HPE ProLiant DL380 Gen10, Procesoare: 2 x Intel Xeon® Silver, 4110 8-Core (2.10GHz 11MB), Memorie 256GB DDR4 2666MHz RDIMM, SSD: 2 x 240GB SATA, HDD: 6 x 1.2TB 12G 10k rpm, 4 porturi Gigabit pentru lansarea în execuţie a infrastructurii necesară derulării unui exerciţiu;
- 1 switch Cisco SG350XG-2F10 12-port 10GBase-T pentru asigurarea redundanţei şi a unei conexiuni de 2 x 10Gbps între servere, echipamentul de stocare şi celelalte componente ale infrastructurii;
- 1 firewall Cisco ASA 5508-X pentru asigurarea securităţii infrastructurii.

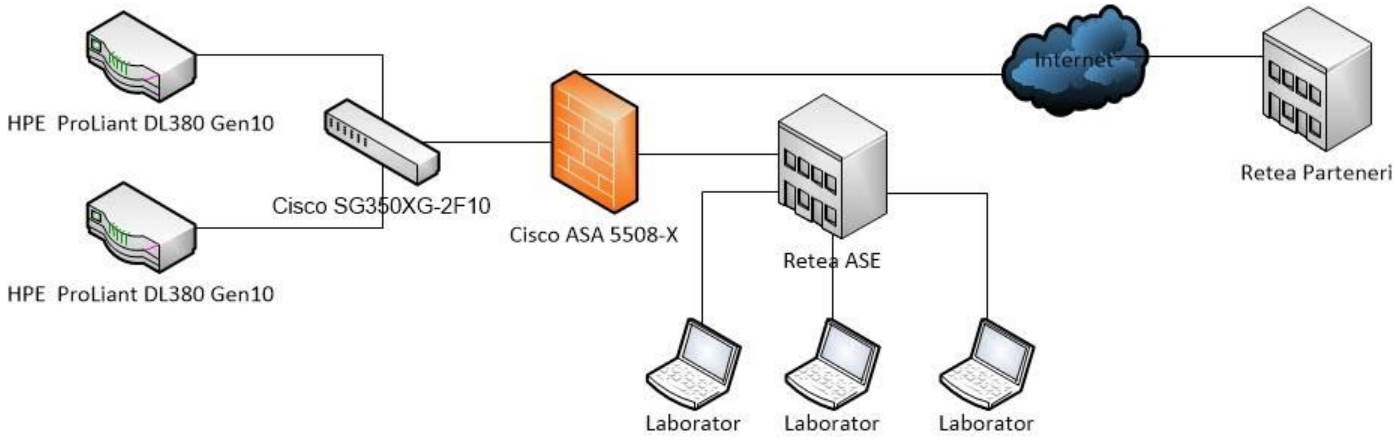

Figura 3.5.7. Topologie retea cyber range ASE

Gestiunea platformei cyber range se face prin intermediul clienţilor de tip vSphere Hypervisor (ESXi) ce au fost instalaţi pe cele 2 servere şi a soluţiei vCenter Server. Astfel, cele 2 servere pot fi gestionate împreună şi resursele agregate pot fi puse la dispoziţia diferitelor scenarii.

După o testare pe componente funcţionale şi servicii dezvoltate în diferite scenarii de utilizare, platforma cyber range a fost folosită pentru derularea de laboratoare la următoarele cursuri de securitate cibernetică din cadrul programului de Securitatea Informatica (www.ism.ase.ro), cu predare în limba Engleză:

- Distributed and Parallel Systems Security;
- Web and Cloud Applications Security
- IoT (Internet of Things) & Embedded Operating Systems Security
- Privacy and Ethical Hacking

Testarea platformei a evidenţiat faptul că pot fi derulate scenarii de nivel simplu sau mediu cu 30-40 de utilizatori ce folosesc resursele virtuale în mod simultan. Problemele întâlnite s-au rezumat la bandwidth-ul maxim disponibil în reţeaua internă ASE şi la viteza de încărcare a maşinilor virtuale, datorată lipsei unei soluţii dedicată pentru stocare de date.

Pe parcursul derulării proiectului, în cadrul ASE au fost elaborate 3 scenarii pentru exerciţii de apărare cibernetică, astfel:

Scenariul 1 ASE - Atac de tip DoS asupra protocolului MQTT folosit în arhitecturi IoT

MQ Telemetry Transport (MQTT) este un protocol de mesagerie ce foloseşte un mecanism de publicare / abonare. Iniţial a fost proiectat de Andy Stanford-Clark şi Arlen Nipper şi este în prezent un standard OASIS (Organization for the Advancement of Structured Information Standards). În prezent, protocolul MQTT reprezintă un standard definit în ISO / IEC 20922: 2016 ((Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1). Acest protocol este utilizat pe scară largă pentru sisteme IoT caracterizate de resurse limitate deoarece: este un protocol simplu, uşor de implementat şi cu lăţime de bandă mică. Protocolul MQTT nu este unul securizat prin design şi din acest motiv este vulnerabil la diferite tipuri de atacuri.

Acest scenariul este compus dintr-o maşină virtuală Linux Ubuntu cu minim 1vCPU şi 1GB memorie RAM pe care este instalat un broker MQTT Eclipse Mosquitto (https://mosquitto.org/). Prin intermediul unei conexiuni IP (Ethernet sau Wi-Fi) este conectat la brokerul MQTT unul sau mai mulţi clienţi care se abonează la unul sau mai multe canale/subiecte de mesaje. Pentru clienţi se poate folosi orice soluţie din lista https://mqtt.org/software/

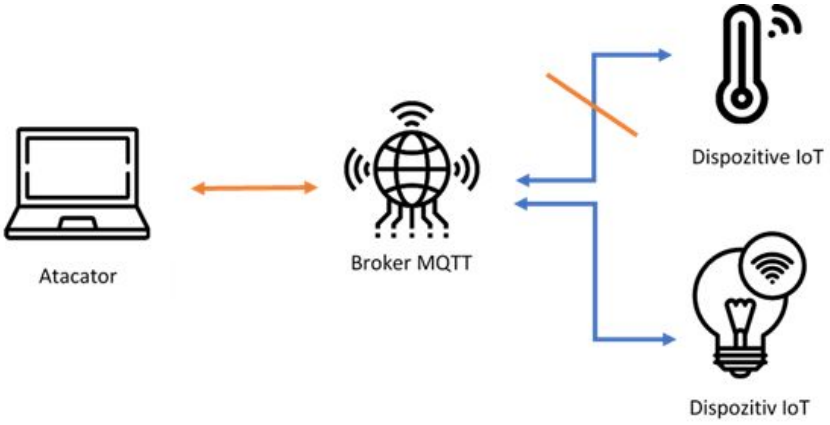

Figura 3.5.8. Scenariu 1 ASE - Atac de tip DoS asupra unui broker MQTT

Obiectivele acestui scenariu sunt:

1. Înţelegerea protocolului MQTT
2. Crearea unui atac de tip Denial of Service (DoS) sau Distributed Denial of Service (DDoS)
3. Folosirea unui client MQTT sau a unei librării pentru dezvoltarea unui client
4. Evaluarea consecinţelor unui atac de tip DoS

Etapele pentru finalizarea cu succes a scenariului:

1. Înţelegerea protocolului MQTT prin prisma modului în care clienţii se abonează şi publică mesaje pe anumite canale
2. Abonarea unui client la un canal şi monitorizarea datelor primite de la brokerul MQTT
3. Dezvoltarea unui client MQTT sau utilizarea unuia existent care va juca rolul atacatorului
4. Publicarea unui număr mare de mesaje pe anumite canale care va forţa brokerul să retrimită mesajele către clienţii existenţi
5. Deschiderea unui număr mare de conexiuni către broker şi menţinerea acestora în starea deschisă
6. Monitorizarea comportamentului broker-ului MQTT prin prisma clientului de la pasul 2
7. Creşterea numărului de mesaje sau conexiuni până în punctul în care clientul de la pasul 2 nu se mai poate conecta la broker şi nu mai primeşte mesaje

Scenariul 2 ASE - Atac de tip blackhole asupra unei topologii RPL pentru a izola un nod sau o parte a reţelei

RPL (Routing Protocol for Low-Power and Lossy Networks) este un protocol de rutare pentru reţelele wireless cu consum redus de energie şi, în general, vulnerabil la pierderi de pachete (lossy). Este un protocol optimizat pentru comunicatii multi-hop şi many-to-one, dar acceptă şi mesaje one-to-one. [https://tools.ietf.org/html/rfc6550]

RPL creează o topologie similară unui arbore (DAG sau grafic aciclic direcţionat). Fiecare nod din reţea are un rang atribuit, care creşte pe măsură ce nodul se îndepărtează de nodul rădăcină. Nodurile retrimit pachetele către alte noduri folosind intervalul cel mai mic ca şi criteriu de selecţie a rutei.

Protocolul RPL este vulnerabil la diferite tipuri de atacuri care afectează conţinutul mesajelor transmite în reţea, topologia acesteia şi canalul de comunicaţie (https://hal.inria.fr/hal-01207859/document)

Un atac al topologiei RPL poate conduce la izolarea unui nod sau a unui subarbore ce are ca rădăcină nodul respectiv. Astfel nodul/subreteaua este izolată de restul topologiei şi mesajele transmise de nodurile izolate nu vor mai ajunge în restul reţelei.

Acest scenariul este compus dintr-o maşină virtuală Instant Contiki (https://sourceforge.net/projects/contiki/files/Instant%20Contiki/) cu minim 1vCPU şi 1GB memorie RAM pe care este instalat simulatorul Cooja (https://anrg.usc.edu/contiki/index.php/Cooja_Simulator).

Reţeaua definită pentru acest scenariu are 12 noduri şi este încărcată în simulatorul Cooja prin intermediul unui fişier .csc dat.

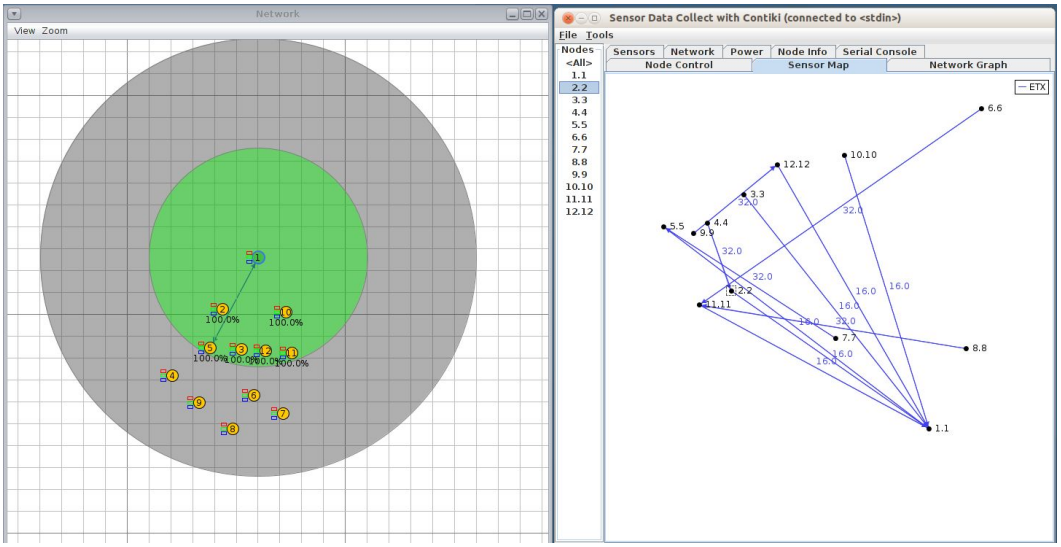

Figura 3.5.9. Scenariu 2 ASE - Topologie reţea RPL cu 12 noduri folosită pentru a simula scenariul

Din diagrama anterioară se poate observa că o serie de noduri IoT, 4,6,7,8 şi 9, sunt în afara razei de transmisie a nodului 1. Din această perspectivă, ele depind de celelalte noduri, 5, 3 12 şi 11, pentru a primi sau transmite mesaje în reţea.

Acest lucru facilitează atacul de tipul blackhole asupra topologiei RPL prin care un atacator preia controlul unui dispozitiv aflat la graniţa reţelei pentru a opri retransmiterea pachetelor primite către alte noduri care depind de acesta.

Obiectivele acestui scenariu sunt:

1. Înţelegerea protocolului RPL
2. Crearea unui atac de tip blackhole asupra unei topologii RPL formată din mai multe noduri IoT
3. Înţelegerea sistemului de operare Contiki destinat dispozitivelor IoT
4. Folosirea unui simulator pentru reţele IoT
5. Modificarea codului sursă al protocolului IP6 al unui client IoT pe care rulează Contiki OS (https://github.com/contiki-os/contiki/blob/master/core/net/ipv6/uip6.c)
6. Evaluarea consecinţelor unui atac de tip blackhole asupra topologiei RPL

Etapele pentru finalizarea cu succes a scenariului:

1. Înţelegerea protocolului RPL prin prisma modului în care clienţii IoT descoperă reţeaua şi definesc poziţia fiecărui nod în topologia RPL
2. Înţelegerea atacurilor de tip blackhole într-o reţea
3. Înţelegerea sistemului de operare Contiki şi a modului în care sunt încărcate diferite module
4. Încărcarea scenariului RPL în simulatorul Cooja ce rulează în maşina virtuală Contiki
5. Utilizarea simulatorului Cooja pentru a rula simularea şi pentru a vizualiza topologia RPL
6. Înţelegerea codului sursă aferent modului IP6 folosit de nodurile din reţeaua RPL pentru a ruta pachete
7. Modificarea codului sursă al modului IP6 pentru a dezactiva retransmiterea mesajelor primite de nodul curent dar care au ca destinaţie alte noduri
8. Recompilarea modului IP6 modificat şi încărcarea acestuia pe un nod IoT care este atacat
9. Rularea simulării pentru a observa efectul atacului de tip blackhole asupra topologiei.

Dacă atacul reuşeşte, topologia din simulator ar trebuii să nu includă o parte din nodurile iniţiale.

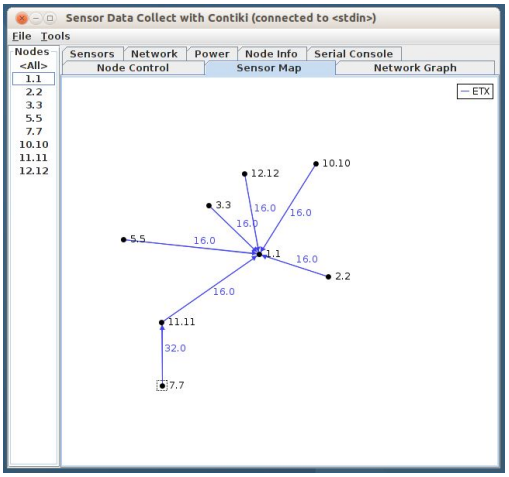

Figura 3.5.10. Scenariu 2 ASE - Topologia RPL după atacul de tip blackhole pe unul din nodurile 5, 3 12 şi 11

Scenariul 3 ASE - Atac de tip flooding asupra unei topologii RPL pentru a izola un nod sau un subarbore

Acest scenariu exploatează vulnerabilităţi ale protocolului RPL (Routing Protocol for Low-Power and Lossy Networks) şi ale reţelelor IoT formate pe baza acestuia. Scenariul este compus dintr-o maşină virtuală Instant Contiki (https://sourceforge.net/projects/contiki/files/Instant%20Contiki/) cu minim 1vCPU şi 1GB memorie RAM pe care este instalat simulatorul Cooja (https://anrg.usc.edu/contiki/index.php/Cooja_Simulator).

Reţeaua definită pentru acest scenariu are 12 noduri şi este încărcată în simulatorul Cooja prin intermediul unui fişier .csc dat.

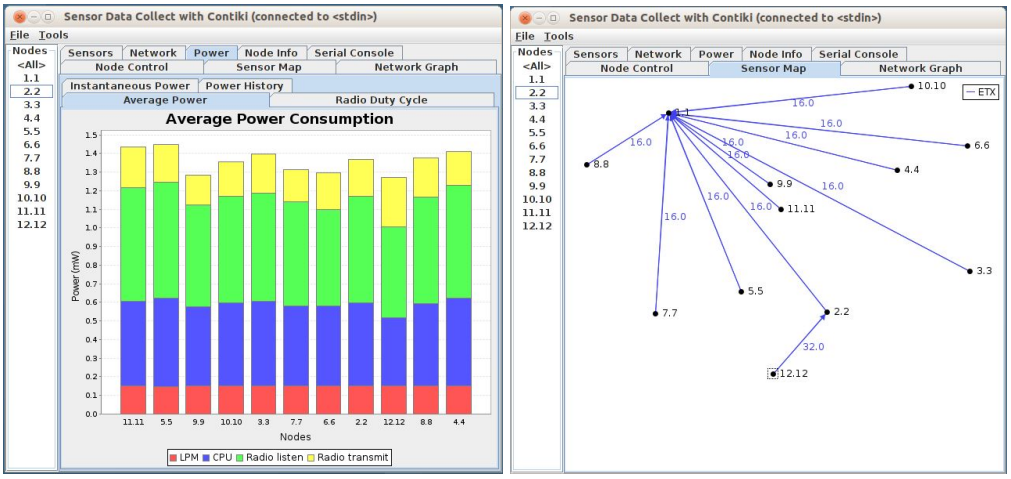

Figura 3.5.11. Scenariu 3 ASE - Topologie reţea RPL cu 12 noduri folosită pentru a simula scenariul

Pentru a comunica între ele, nodurile realizează conexiuni directe şi fiecare nod poate transmite mesaje către alte noduri sau poate retransmite mesaje primite de la alte noduri pentru a le propaga în interiorul topologiei. Fiecare nod este participant activ în topologie şi poate avea 3 roluri: iniţiator de mesaje, destinaţie pentru anumite mesaje sau intermediar pentru mesaje transmise de alte noduri. Acest lucru facilitează atacul de tipul flooding asupra topologiei RPL din orice nod. Prin acest tip de atac un atacator preia controlul unui dispozitiv aflat în topologia RPL şi va genera un flux mare de mesaje care vor supraîncărca nodurile direct adiacente sau o parte a topologiei. Ca rezultat al atacului, o parte din noduri nu vor mai putea procesa mesaje valide şi astfel reţeaua nu mai poate funcţiona. Atacul conduce la un Denial of Service (DoS) asupra nodurilor adiacente sau asupra unei părţi din reţea.

Obiectivele acestui scenariu sunt:

1. Înţelegerea protocolului RPL
2. Crearea unui atac de tip flooding asupra unei topologii RPL formată din mai multe noduri IoT
3. Înţelegerea sistemului de operare Contiki destinat dispozitivelor IoT
4. Folosirea unui simulator pentru reţele IoT
5. Modificarea codului sursă al protocolului RPL al unui client IoT pe care rulează Contiki OS ()
6. Exploatarea protocolului RPL prin prisma mesajelor de tip DODAG Information Solicitation (DIS), https://tools.ietf.org/html/rfc6550#page-38
7. Evaluarea consecinţelor unui atac de tip *flooding* asupra topologiei RPL

Etapele pentru finalizarea cu succes a scenariului:

1. Înţelegerea protocolului RPL prin prisma modului în care clienţii IoT descoperă reţeaua şi definesc poziţia fiecărui nod în topologia RPL
2. Înţelegerea mesajelor de tip DODAG Information Solicitation (DIS), https://tools.ietf.org/html/rfc6550#page-38
3. Înţelegerea atacurilor de tip flooding într-o reţea RPL
4. Înţelegerea sistemului de operare Contiki şi a modului în care sunt încărcate diferite module
5. Încărcarea scenariului RPL în simulatorul Cooja ce rulează în maşina virtuală Contiki
6. Utilizarea simulatorului Cooja pentru a rula simularea şi pentru a vizualiza topologia RPL
7. Înţelegerea codului sursă aferent modului RPL şi a parametrilor specifici mesajelor de tip DIS
8. Modificarea codului sursă al modului RPL pentru a modifica modul în care sunt transmise mesajele de tip DIS pentru a genera atacul de tip *flooding* (https://github.com/contiki-os/contiki/blob/master/core/net/rpl/rpl-timers.c şi https://github.com/contiki-os/contiki/blob/master/core/net/rpl/rpl-private.h)
9. Recompilarea modulului RPL modificat şi încărcarea acestuia pe un nod IoT care este atacat
10. Rularea simulării pentru a observa efectul atacului de tip *flooding* asupra topologiei.

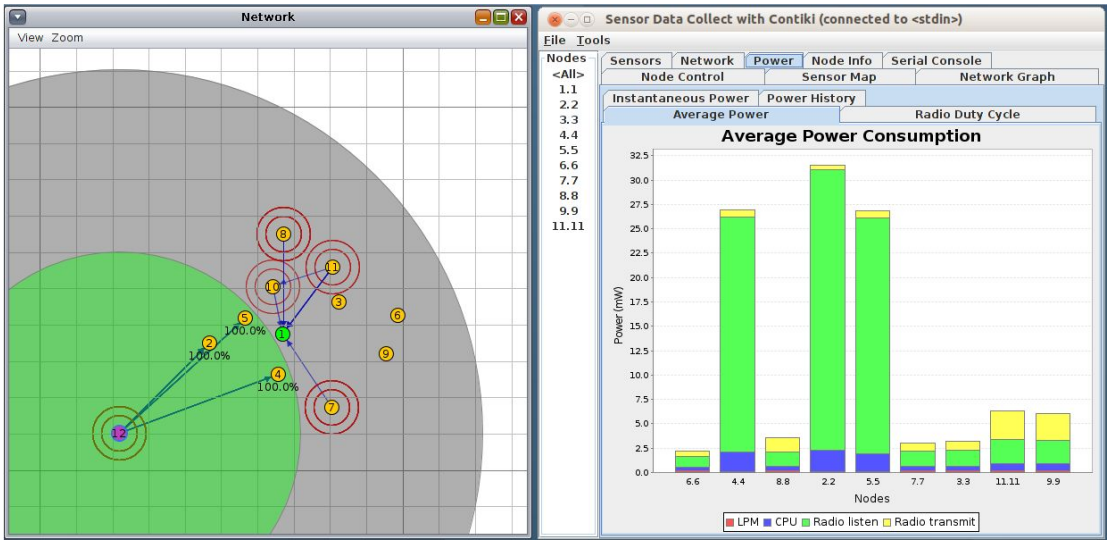

Figura 3.5.12. Scenariu 3 ASE - Topologia RPL după atacul de tip flooding asupra nodului 12

Dacă atacul reuşeşte, nodurile direct conectate la nodul atacat vor afişa valori foarte mari pentru consumul de electricitate deoarece nodurile respective vor procesa un număr mare de cereri. Astfel, nodurile nu vor mai putea participa activ în topologia RPL.

Universitatea din Bucureşti (UB)

La Universitatea din Bucureşti, platforma cyber range a fost instalată în cadrul Data Center-ului administrat de Direcţia IT a universităţii. Infrastructura hardware folosită pentru instalare a fost formată din echipamente existente în cadrul centrului sau achiziţionate în cadrul proiectului. Echipamentele sunt de ultimă generaţie, utilizate pe larg în mediul academic şi în industria de profil:

- 1 server DELL PowerEdge R530: Procesoare: 20 CPUs x Intel Xeon E5-2630 v4 @ 2.20 GHz; Memorie 256 GB RAM; HDD 1.82 TB, controller RAID Avago (LSI) Dell PERC H730 Mini; Reţea: 4 x Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet; Placă video Matrox Electronics Systems Ltd. G200eR2
- 1 workstation ThinkStation P620 AMD Threadripper: Procesor AMD Threadripper Pro 3955WX (16-cores, 8MB Cache, up to 4.30GHz); Memorie 64GB DDR4 3200MHz RDIMM QTY(2); Placa video NVIDIA Quadro P620 2GB (4xMini DP) High Profile; RAID 1; SDD 1 TB
- 1 switch Cisco Catalyst 3850
- 1 firewall Cisco ASA 5508 pentru asigurarea securităţii infrastructurii.

Gestiunea platformei cyber range se face prin intermediul clienţilor de tip vSphere Hypervisor (ESXi) ce au fost instalaţi pe server şi a soluţiei vCenter Server.

Platforma cyber range va fi folosită pentru derularea de laboratoare la următoarele cursuri de securitate cibernetică din cadrul programului de master “Securitate şi Logica Aplicată” (https://sla.cs.unibuc.ro), cu predare în limba engleză, dar şi la cursuri de la licenţă:

- Cybersecurity
- Network Security
- Operating systems: design and security
- Special topics in Security and Applied Logic
- Modern Technologies for Information security
- Securitatea fluxului informaţional
- Securizarea şi automatizarea reţelelor
- Criptografie si securitate.

Următoarele scenarii au fost elaborate pentru Universitatea din Bucureşti:

Scenariul 1 UB - Analiza manuala a unui malware

În acest scenariu studenţii vor realiza analiza manuala a unui malware de tip ransomware, scopul fiind atat familiarizarea cu modul de construcţie a unui malware, cât şi cu uneltele folosite pentru a realiza analize statice sau dinamice. Scenariul a fost construit pentru a rula folosind o masina virtuala Windows cu uneltele necesare analizei, fiind nevoie de aproximativ 8 GB RAM şi 2 vCPU.

Obiectivele acestui scenariu sunt:

1. Investigarea unui malware folosind instrumentele Windows Internals
2. Analiza traficului de reţea generat de un malware
3. Analiza fisierelor create de un malware
4. Analiza modificarilor realizate de un malware asupra registrelor Windows
5. Modificarea unui malware şi analiza comportamentala a soluţiilor de tip antivirus

Etapele pentru finalizarea cu succes a scenariului:

1. Identificarea fişierului care este cu potenţial malware folosind o arhiva de început cu 10 fişiere similare
2. Generarea unui hash a tuturor fişierelor şi analizarea lor cu platforma online www.virustotal.com

3. Folosirea Wireshark pentru analiza traficului generat de către malware. De observat care sunt adresele DNS folosite de către malware.
4. Monitorizarea şi inspectarea conexiunilor TCP active. Identificarea modului de conectare la Internet prin intermediul reţelei TOR.
5. Utilizarea programului Procmon pentru a identifica toate intrările în regiştrii Windows.
6. Identificarea intrării în regiştrii Windows prin care malware-ul configurează pornirea automată la startup
7. Monitorizarea fişierelor create sau modificate de către malware.
8. Folosirea programului ResourceHacker pentru a modifica malware-ul astfel incat valoarea hash sa difere.
9. Descoperirea portofelului electronic folosit de către malware în reţeaua Bitcoin.

Scenariul 2 UB - Analiza automată a unui malware

În acest scenariu studenţii vor realiza analiza automată a unui malware de tip ransomware, scopul principal fiind familiarizarea cu unelte de tip sandboxing şi analiza rapoartelor generate de către acestea. Pentru acest scenariu este necesara o masina virtuala cu minim 16 GB şi 4 vCPU, care permite virtualizarea resurselor (“nested virtualization”).

Scenariul are preconfigurata o instanta de sandboxing (https://cuckoosandbox.org/) care poate rula un malware într-o masina virtuala de Windows 7. De menţionat ca maşina de analiza folosită de către CuckooSandbox are dezactivate toate mecanismele de securitate implicite, precum firewall sau antivirus, iar accesul la Internet este de asemenea restrictionat.

Obiectivele acestui scenariu sunt:
1. Investigarea unui malware folosind analiza automată
2. Detonarea unui malware automată
3. Analiza rapoartelor generate de catre CuckooSandbox
4. Crearea unei semnături pentru a identifica malware-ul analizat

Etapele pentru finalizarea cu succes a scenariului:
1. Analiza manuala a tuturor fisierelor din arhiva pusa la dispozitie (numele, data la care au fost create, etc.)
2. Familiarizarea cu CuckooSandbox şi analiza unui singur fişier folosind platforma web.
3. Analizat tuturor fişierelor folosind API-ul pus la dispoziţie de către CuckooSandbox.
4. Descoperirea notelor pentru fiecare executabil ce trebuie testat si identificarea malware-ului.
5. Analiza raportului generat pentru malware şi descoperirea tuturor fişierelor create, conexiunilor de reţea deschise, intrărilor noi apărute în regiştri Windows.
6. Scrierea unei reguli Yara pentru a identifica malware-ul.
7. Analiza malware-ului folosind CuckooSandbox şi a regulii create anterior.

Scenariul 3 UB - Analiza metodelor de criptare a datelor

În acest scenariu studenţii vor folosi unelte criptografice pentru a transmise mesaje confidenţiale. Pentru acest scenariu se vor folosi 3 maşinii virtuale, una care furnizeaza accesul la Internet iar 2 pentru realizarea schimbului de mesaje criptate.

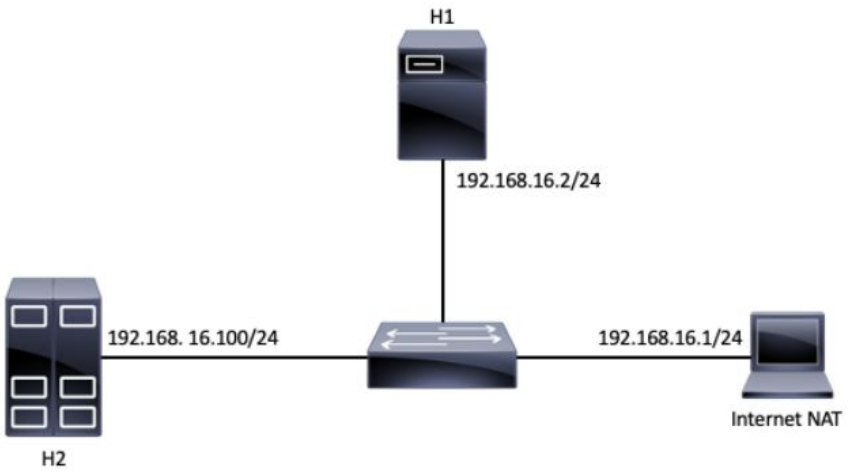

Figura 3.5.13. Scenariul 3 UB - Analiza metodelor de criptare a datelor

Obiectivele acestui scenariu sunt:

1. Familiarizarea cu metodele de criptare simetrică
2. Familiarizarea cu metodele de criptare asimetrica
3. Înţelegerea metodelor de protejare a integrităţii prin hashing
4. Înţelegerea formatului Base64

Etapele pentru finalizarea cu succes a scenariului:

1. Folosirea diferiţilor algoritmi de hash (MD5, SHA1, SHA2) pentru a oferi integritatea datelor.
2. Înţelegerea conceptului de HMAC şi care este utilitatea lui atunci cand datele se transmit pe reţea.
3. Criptarea simetrică folosind utilitarul openssl.
4. Transmiterea unui mesaj criptat simetric prin intermediul unui email şi codificarea sa în format Base64
5. Înţelegerea modurilor de criptare simetrică (stream vs block) şi a tipurilor de criptare simetrică (ECB, CBC, etc).
6. Atacuri asupra modului de criptare ECB prin vizualizarea unei imagini criptate în acest mod.
7. Semnarea unui fişier folosind algoritmul DSA.
8. Verificarea semnăturii folosind algoritmul DSA.
9. Criptarea asimetrica şi transmiterea de mesaje criptate folosind unealta openssl si algoritmul RSA.

Diseminarea rezultatelor proiectului

Rezultatele activităţilor de cercetare derulate în cadrul proiectului au fost publicate în jurnale şi în volumele unor conferinţe internaţionale indexate ISI/BDI. În total, au fost publicate **21 articole ştiinţifice**, după cum urmează:

1. C. Lupascu, A. Lupascu, and I. Bica, “DLT Based Authentication Framework for Industrial IoT Devices,” Sensors, vol. 20, no. 9, p. 2621, May 2020;
2. I. Aciobanitei, I. Luculescu, and M.-L. Pura, “Qualified Electronic Signature SaaS Solution for Google Docs & Google Sheets Documents,” 2020 13th International Conference on Communications (COMM), Jun. 2020;
3. I. Aciobanitei, V. Dedita, M.-L. Pura, and V.-V. Patriciu, “SABRES - A Proof of Concept for Enhanced Cloud Qualified Electronic Signatures,” 2020 13th International Conference on Communications (COMM), Jun. 2020;

4. E.-C. Ruica, M.-L. Pura, and I. Aciobanitei, “Implementing Cloud Qualified Electronic Signatures for Documents using Available Cryptographic Libraries: A Survey,” 2020 13th International Conference on Communications (COMM), Jun. 2020;
5. L. Deshotels, C. Carabas, J. Beichler, R. Deaconescu, and W. Enck, “Kobold: Evaluating Decentralized Access Control for Remote NSXPC Methods on iOS,” 2020 IEEE Symposium on Security and Privacy (SP), May 2020;
6. T.-M. Georgescu, B. Iancu, A. Zamfiroiu, M. Doinea, C. E. Boja, and C. Cartas, “A Survey on Named Entity Recognition Solutions Applied for Cybersecurity-Related Text Processing,” Advances in Intelligent Systems and Computing, pp. 316–325, Oct. 2020;
7. C. Toma, M. Popa, M. Doinea, “A.I. neural networks inference into the IoT embedded devices using TinyML for pattern detection within a security system,” Proceedings of the IE 2020 International Conference, May 2020;
8. A. Zamfiroiu, B. Iancu, C. Boja, T.-M. Georgescu, C. Cartas, M. Popa, and C. V. Toma, “IoT Communication Security Issues for Companies: Challenges, Protocols and The Web of Data,” Proceedings of the International Conference on Business Excellence, vol. 14, no. 1, pp. 1109–1120, Jul. 2020;
9. T.-M. Georgescu, “Natural Language Processing Model for Automatic Analysis of Cybersecurity-Related Documents,” Symmetry, vol. 12, no. 3, p. 354, Mar. 2020;
10. C. Paduraru, M. Paduraru, and A. Stefanescu, “Optimizing decision making in concolic execution using reinforcement learning,” 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Oct. 2020;
11. M. Cernat, A.-N. Staicu, and A. Stefanescu, “Towards Automated Testing of RPA Implementations”, In Proc. of A-TEST’20, workshop affiliated to ESEC/FSE'20, pp. 21-24, ACM, 2020;
12. A. Pătraşcu, C. Păduraru, and P. Irofti, “Stochastic SPG with Minibatches,” Studies in Computational Intelligence, pp. 3–25, Sep. 2020;
13. C. I. Paduraru and G. Stefanescu, “Adaptive Virtual Organisms: A Compositional Model for Complex Hardware-software Binding”, Fundamenta Informaticae, vol. 173, no. 2–3, pp. 139–176, Mar. 2020;
14. C. Paduraru, B. Ghimis, and A. Stefanescu, “RiverConc: An Open-source Concolic Execution Engine for x86 Binaries,” Proceedings of the 15th International Conference on Software Technologies, 2020;
15. M. Cernat, A.-N. Staicu, and A. Stefanescu, “Improving UI Test Automation using Robotic Process Automation,” Proceedings of the 15th International Conference on Software Technologies, 2020;
16. P. Irofti, and A. Băltoiu, “Unsupervised dictionary learning for anomaly detection,” To appear in Proc. of iTWIST’20 (international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques), 2020;
17. A. Băltoiu, A. Pătraşcu, and P. Irofti, “Graph Anomaly Detection Using Dictionary Learning,” To appear in The 21st World Congress of the International Federation of Automatic Control, 2020, pp. 1-8;
18. C. Paduraru, A. Stefanescu, and B. Ghimis, “Testing Multi-tenant Applications using Fuzzing and Reinforcement Learning“. In Proc. of LANGETI'20, workshop affiliated to ESEC/FSE'20, pp. 1-6, ACM, 2020;
19. B. Ghimis, M. Paduraru, and A. Stefanescu, ”RIVER 2.0: An Open-Source Testing Framework using AI Techniques,”; In Proc. of LANGETI'20, workshop affiliated to ESEC/FSE'20, pp. 13-18, ACM, 2020;

20. A.-E. Panait, R. F. Olimid, and A. Stefanescu, ”Identity Management on Blockchain - Privacy and Security Aspects”, Proceedings of the Romanian Academy, Series A, Volume 21, Number 1/2020, pp. 45-52, 2020;
21. A.-E. Panait, R. F. Olimid, and A. Stefanescu, “Analysis of uPort Open, an Identity Management Blockchain-Based Solution,” Lecture Notes in Computer Science, pp. 3–13, 2020.

Concluzii

Având în vedere cele expuse mai sus, apreciem că au fost atinse toate obiectivele manageriale şi ştiinţifice pentru această etapă. Indicatorilor de rezultat prevăzuţi au fost realizaţi în întregime şi chiar depăşiţi la unele capitole. Rezultatele cercetărilor efectuate au fost diseminate prin publicarea de articole în jurnale şi la conferinţe de prestigiu. În total, în această etapă au fost publicate 21 articole faţă de 4 estimate iniţial.

Proiectul component 1 (*Metode formale şi tehnici de învăţare automată pentru securizarea aplicaţiilor şi sistemelor de operare*) şi proiectul component 2 (*Arhitecturi de securitate pentru IoT*) s-au încheiat odată cu această etapă, indicatorii stabiliţi iniţial fiind realizaţi în totalitate. Astfel, în cadrul proiectului component 1, au fost realizate 2/2 studii tehnice, au fost dezvoltate 2/2 produse noi (*Platforma online pentru analiza binarelor, Framework de securitate bazat pe metode formale şi învăţare automată*) şi au fost publicate 28/5 articole ştiinţifice în jurnale / conferinţe indexate BDI. În cadrul proiectului component 2, au fost realizate 2/2 studii tehnice, 2/2 produse noi (*Metodologie pentru evaluarea securităţii IoT, Framework de securitate bazat pe reputaţie pentru IoT*) şi au fost publicate 22/5 articole ştiinţifice în jurnale / conferinţe indexate BDI. Totodată, pe baza echipamentelor achiziţionate, au fost create, la nivelul universităţilor partenere, laboratoare noi pentru identificarea vulnerabilităţilor în aplicaţii software şi sisteme de operare sau pentru dezvoltarea de soluţii IoT.

Proiectul component 3 (*Mediu distribuit pentru exerciţii şi cercetare avansată în domeniul securităţii cibernetice*) va continua până în Septembrie 2021 cu achiziţia ultimului lot de echipamente şi testarea platformei cyber range federate prin organizarea în comun a unui exerciţiu de apărare cibernetică la care vor fi invitate să participe şi alte universităţi din ţară care organizează cursuri sau desfăşoară cercetări în domeniul securităţii cibernetice.

Data: 08.12.2020

Director Proiect Complex

Prof.univ.dr.ing.

Ion BICA