UNIVERSITATEA POLITEHNICA BUCUREȘTI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

|  |  |
| --- | --- |
| upb | cs |

PROIECT DE DIPLOMĂ

Embedded Devices Malware Detection, Prevention & Centralization

Andrei Grigoras

**Coordonator științific:**

Prof. dr. ing. Cornel Popescu

BUCUREŞTI

2021

# Sinopsis

Scopul acestui proiect a fost de a implementa un mecanism rapid, portabil, ușor de instalat, si gratis de detecție a fișierelor malițioase ce pot apărea pe un sistem **Unix** ca urmare a descărcării de pe Internet sau de pe un dispozitiv extern. Astfel, am urmărit crearea unei soluții de detectare in limbajul **Bash** ce este nativ tuturor sistemelor cu sistem de operare **Unix**, care va servi in implementarea unei arhitecturi client-server (in cazul nostru, mai mulți clienți, un singur server).

In urma unei investigări pentru a găsi software echivalent (gratis si disponibil pentru orice sistem Unix), am observat absenta unei astfel de soluții. Din acest motiv, proiectul se adresează dispozitivelor embedded din categoria „Internet of Things”, si urmărește monitorizarea unei rețele de acest tip de dispozitive (clienți IoT) prin intermediul unui software ce va detecta si carantina fișierele malițioase de pe sistem. Serverul va permite, printr-o aplicație web, colectarea si centralizarea alertelor la nivel de rețea.

Cuvinte cheie: **malware**, **Unix**, **gratis**, **embedded devices**, **IoT**, **rețea**, **detecție** si **prevenire**.

# Abstract

The goal of this project was to implement a fast, portable, easy to install, and free mechanism which can detect malicious files that enters a **Unix** system via Internet (download) or external devices (USB, CD). Therefore, I looked for creating a solution using **Bash** scripting language, which is the native language used my **Unix** systems, that can serve in implementing a client-server architecture (multiple clients, one server).

After a thorough investigation to find similar software (free and deployable on any Unix system), I noticed that the market lacks of any such solution. Therefore, the project targets embedded devices such as the one used in “Internet of Things” and performs continuous monitoring of a network of such devices. This is done via a software that detects and monitors any newly malicious file on the systems as well as a web application that is deployed on the server which centralizes all alerts within the network.

Keywords: **malware**, **Unix**, **free**, **embedded devices**, **IoT**, **network**, **detection** and **prevention**.

**CUPRINS**

[Sinopsis 1](#_Toc71844368)

[Abstract 1](#_Toc71844369)

[1 Introducere 3](#_Toc71844370)

[1.1 Context 3](#_Toc71844371)

[1.2 Problema 3](#_Toc71844372)

[1.3 Obiective 3](#_Toc71844373)

[1.4 Structura lucrării 3](#_Toc71844374)

[2 Analiza și specificarea cerințelor 4](#_Toc71844375)

[3 Studiu de piață / Abordări existente 5](#_Toc71844376)

[4 Soluția propusă 6](#_Toc71844377)

[5 Detalii de implementare 7](#_Toc71844378)

[6 Studiu de caz / Evaluarea rezultatelor 8](#_Toc71844379)

[7 Concluzii 9](#_Toc71844380)

[8 Bibliografie 10](#_Toc71844381)

[9 Anexe 11](#_Toc71844382)

# Introducere

## Context

Dorința implementării acestei soluții a venit ca urmare a unei curiozități de creare a unui software de tip **antivirus**, folosind doar resurse gratuite de tipul **open-source** care, pe baza unui fișier încărcat, determina daca fișierul este sau nu malițios. Pe de alta parte, proiectul implementat diferă de o soluție clasica de antivirus prin piața ce o urmărește, aceea a dispozitivelor embedded (precum cele din IoT) ce oferă resurse limitate la nivelul sistemului de operare, făcând astfel imposibila instalarea unei soluții precum Avira/Bitdefender/Avast. (ce sunt specifice sistemului de operare Windows).

Deși exista mai multe soluții open-source care pot detecta malware prezent pe un sistem, niciuna din aceste soluții nu pot detecta pasiv si continuu si nici nu oferă o modalitate de a centraliza astfel de alerte la nivel de rețea.

## Problema

Prima problema pe care proiectul o abordează e cea a nevoii unui software de tipul antivirus care sa funcționeze pe orice sistem Unix. O a doua problema este cea financiara întrucât pe piața exista astfel de soluții (mult mai complexe) denumite „Host based Intrusion Detection System” sau „Endpoint detection and response” dar care necesita achiziționarea unui abonament/licențe.

## Obiective

Obiectul principal al proiectului este de a reuși cu succes să detectăm și eradicăm fișierele malițioase noi apărute pe mai multe sistem Unix cat si centralizarea grafica a acestor alerte printr-o aplicație web. Succesul proiectului este determinat de existența unei soluții open-source ce expune un API prin care se pot încărca fișiere care, in urma analizei, sunt detectate sau nu a fi malițioase.

## Structura lucrării

In secțiunea următoarea, vom analiza si specifica cerințele ce au fost luate in considerare in implementarea proiectului cat si modalitățile/abordarea care a garanta îndeplinirea lor.

După, vom face o analiza a pieței, soluțiile existente, cat si factorii prin care soluția propusa se evidențiază in acest domeniu. Vom porni de la contextul atacurilor de tip malware si vom sfârși prin a înțelege utilitatea soluției noastre in industria IoT.

In secțiunile 4 si 5 vom discuta despre soluția propusa (arhitectura, diagrama, cazuri de utilizare) cat si detalii specifice precum bucăți de cod, funcționalități si limitări.

In secțiune 6, vom evalua performantele prin crearea unui mediu de test si simulând un utilizator. Astfel vom putea deduce timpul de răspuns, eficienta dar si corectitudinea datelor.

Ultimele secțiuni sunt adresate unor concluzii, bibliografii folosite cat si anexe utile in înțelegerea soluției.

# Analiza și specificarea cerințelor

In vederea implementării unui proiect care sa aducă un plus industriei IT, acesta trebuie sa îndeplinească anumite cerințe ce sunt direct proporționale cu nivelul curent de evoluției al industriei. Astfel, urmând specificațiile serviciilor de succes din industrie precum Amazon si Google, am compus următoarea lista de cerințe ce trebuie luate in considerare pe parcursul dezvoltării soluției:

* scalabilitate
* disponibilitate
* viteza cat mai ridicata a procesării
* costuri cat mai reduse
* resurse de calcul cat mai scăzute (evitare overhead)
* ușor de instalat si configurat

Abordarea propusa se pliază ușor pe aceste criterii întrucât, folosind un limbaj nativ sistemului de operare ce suporta si multi-threading (Bash), atât criteriul de scalabilitate cat si cel de viteza sunt ușor de îndeplinit. Pentru a facilitata instalarea ușoara a produsului, trebuie realizat un script care, in urma execuției, va realiza toți pașii necesari pregătirii si instalării de software. De asemenea, întrucât produsul final este open-source, am adăugat comentarii pentru fiecare bloc de cod, dar si indicații in cazul in care se dorește configurarea produsului cu alți parametri/opțiuni.

Întrucât scopul proiectului a fost de a crea un produs folosind doar resurse gratuite open-source, costul instalării si folosirii este zero. De asemenea, deoarece procesarea se face prin intermediul soluțiilor open-source ce rulează in **cloud**, resursele necesare rulării sunt minime si sunt strict dependente de calitatea conexiunii la Internet la rețea. (pentru interacțiunea cu API-ul si transmiterea alertelor din rețea către server)

Nu in ultimul rând, pentru a asigura disponibilitatea, produsul a fost testat si remediat de bug-uri/probleme prin emularea unui mediu cu 4 clienți si 1 server. De asemenea, software-ul a fost instalat ca si serviciu ceea ce înseamnă ca, va fi rulat automat atunci când sistemul pornește iar in cazul unei defecțiuni/erori, serviciul se va reporni automat.

Ce trebuie luat in considerare este ca, criteriile enumerate mai sus nu au fost prioritatea numărul unu, motiv pentru care, o abordare folosind alte tehnologii ar fi putut sa îmbunătățească performantele generale ale produsului final. Acest lucru este datorat faptului ca, prioritatea numărul a fost crearea unei soluții care sa poată detecta fișiere malițioase pe **orice** tip de sistem Unix (chiar daca nu exista **Python**, **PHP**, **SQL** sau un manager de pachete instalat) si sa nu implice niciun cost aferent instalării/rulării. Totuși, s-a ținut cont de criteriile standard pentru un produsul ce poate concura pe piața cu altele din aceeași categorie. (majoritatea fiind mult mai avansate si mai complexe)

# Studiu de piață / Abordări existente

Pentru a putea implementa o soluție cat mai buna si care sa poată fi folosita înafara unui „Proof of Concept”, trebuie sa analizam piața aferenta zonei pe care produsul o adresează, in cazul nostru fiind cea alcătuita din software ce ajuta in detectarea de malware.

In analiza pieței, am consultat tipurile de soluții ce pot concura cu proiectul nostru, acestea împărțind-se in mai multe categorii:

* HIDS (Host Based Intrusion Prevention Systems)
* EDR (Endpoint Detection and Response)
* Tool-uri open-source

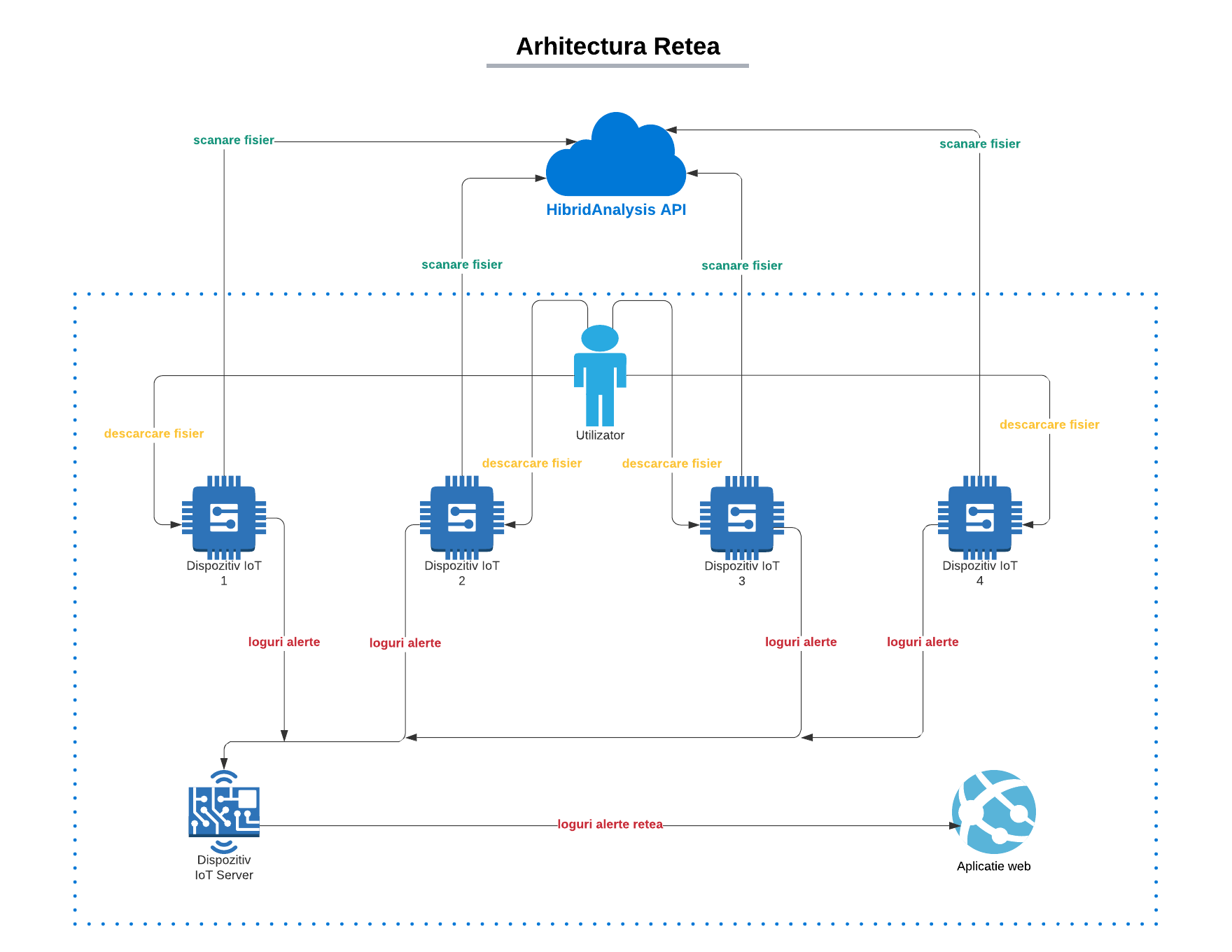
Categoriile de HIDS si EDR sunt adresate mediilor corporate întrucât sunt construite pentru a scala pentru mii de sisteme. De asemenea, acestea încorporează o sumedenie de funcționalități de prevenire si detecție a incidentelor de securitate pe un host (nu doar detecție de malware). Problema acestor soluții este reprezentata, in primul rând, de partea financiara întrucât implica costuri ridicate in funcție de dimensiune rețelei si de pachetul de funcționalități ales. Printre cele mai cunoscute soluții, se numără si SolarWinds, Splunk, RedCloak, Palo Alto, Carbon Black, Fireye. Costul mediu pentru o astfel de soluție este de 50 dolari / sistem / an de unde si necesitatea unei soluții ieftine de detecție malware pentru rețele mici – medii ca si dimensiune. ( < 20 stații)

Pe de alta parte, exista câteva soluții gratis, open-source care pot scana un sistem si detecta diverse tipuri de malware dar si alte tipuri de probleme precum configurații greșite, software cu versiuni ne-update etc. Una din cele mai cunoscute astfel de soluții este Wazuh/OSSEC cu care, personal, am interacționat si am observat funcționalitățile/limitările. Deși aceasta soluție este net superioara celei propuse de noi, ea nu se adresează tuturor sistemelor Unix si depinde de existenta unui „package manager” sau a unei soluții precum **Docker** la nivelul sistemului de operare (pentru a seta mediul de rulare al software-ului). In cazul dispozitivelor embedded ce au un kernel personalizat, aceste dependențe nu pot fi obținute.

Ținând cont de aceste aspecte ale pieței, am considerat ca proiectul va acoperi un mare gap in industrie, adresânduse unei industrii nișate dispozitivelor embedded care. Acestea, in multe circumstanțe, doresc sa fie construite cu resurse cat mai puține dar sa prezinte anumite masuri de siguranța la nivelul securității sistemului. Putem concluzia astfel ca, piața curenta reprezintă o oportunitate pentru un software ca cel propus de proiectul nostru care, deși nu va fi monetizat, va reprezenta un plus pentru industria securității cibernetice pentru dispozitive embedded.

# Soluția propusă

Soluția propusa se adresează unei arhitecturi cu mai mulți clienți ce sunt reprezentate de dispozitive IoT si de un server care este tot un dispozitiv IoT (cel mai practic ar fi o plăcută Raspberry Pi). Software-ul ce rulează pe clienți va încarcă periodic fișierele noi apărute in sistem pe platforma HibridAnalysis folosind API-ul expus de acesta. Daca fișierul este detectat ca fiind malițios, se adaugă intr-un fișier de log care, la rândul sau, este transmis către server la orice noua intrare. Acesta, serverul, are rol in colectarea logurilor de la toți clienții (alerte de fișiere potențial malițioase) si de centralizarea lor printr-o aplicație web ce va afișa informații precum: IP si hostname client, număr alerte in ultimele 24 ore cat si un sumar al acestor fișiere. Astfel, arhitectura poate fi reprezentata sub forma următoarei scheme:

*Figura 1. Arhitectura retelei propuse*

Soluția propusa nu are nicio dependință sau componenta hardware întrucât este reprezentata de o multitudine de servicii (software) care trebuie instalate atât pe clienți cat si pe un dispozitiv ce va servi ca si server. Corectitudinea soluției este strâns data de 2 factori:

* corectitudinea detecției soluției oferite de HibridAnalysis prin API-ul expus
* transmiterea corecta si in timp a datelor de la client la API si la server (sa nu fie corupte sau sa nu ajungă deloc)

# Detalii de implementare

De completat

# Studiu de caz / Evaluarea rezultatelor

De completat

# Concluzii

De completat

# Bibliografie

De completat

# Anexe

De completat