Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Server pre systém na detekciu indikátorov kompromitácie

Bakalárska práca

2016
Michal Fikar

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Server pre systém na detekciu indikátorov kompromitácie

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Jaroslav Janáček PhD.

Bratislava, 2016
Michal Fikar

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Fikar
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Server pre systém na detekciu indikátorov kompromitácie
Server for Indicators of Compromise Detection System

Cieľ: Cieľom práce je navrhnúť a implementovať server pre správu a distribúciu indikátorov kompromitácie a správu výsledkov kontroly prítomnosti týchto indikátorov na veľkom množstve koncových počítačov. Súčasťou práce bude návrh databázy na uchovávanie indikátorov kompromitácie ako výskyt určeného súboru, procesu, certifikátu, kľúča v registroch Windows, prítomnosť sieťovej komunikácie, mutexu, DNS záznamu, alebo ich logickej kombinácie. Databáza bude zároveň obsahovať informácie o výsledkoch kontroly prístomnosti indikátorov kompromitácie na koncových systémoch. Server bude umožňovať prostredníctvom webového rozhrania prezeranie, vkladanie a úpravu indikátorov, import a export indikátorov zo/do súborov v určených formátoch a prezeranie a vyhodnocovanie výsledkov kontrol s možnosťou generovania reportov podľa zadaných kritérií. Server zároveň umožní preberanie definície indikátorov koncovými systémami a automatické zaznamenávanie výsledkov kontrol do databázy.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 28.10.2015

Dátum schválenia: 28.10.2015 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedených zdrojov.

.....

Poďakovanie: Chcel by som sa poďakovať môjmu školiteľovi RNDr. Jaroslavovi Janáčkovi a CSIRT.sk za možnosť pracovať na tomto projekte.

Abstrakt

Indikátory kompromitácie (IOC) sú relatívne nový trend vo vyšetrovaní incidentov v počítačovej bezpečnosti. Umožňujú jednoducho a efektívne opísať stopy zanechané malvérom alebo nepovolenými vniknutiami.

Cieľom tejto práce je vytvoriť server pre systém na detekciu IOC, ktorý bude slúžiť na prehľadávanie počítačov a vyhodnocovanie výsledkov. Úlohou servera v tomto systéme je vytváranie definícií indikátorov, ich distribúcia ku klientským aplikáciám a následné zbieranie a ukladanie výsledkov kontrol.

Server sa skladá z webovej služby na komunikáciu s klientmi, vytvorenej v jazyku PHP a webovej aplikácie na správu, vytvorenej pomocou frameworkov AngularJS a Bootstrap.

Kľúčové slová: indikátory, kompromitácie, IOC, server

Abstract

Indicators of compromise (IOC) are a relatively new trend in investigation of computer security incidents. They can be used to easily and effectively describe traces left behind by malware or unauthorized intrusions.

The goal of this thesis is to create a server for IOC detection system, which will be used to search computers and evaluate the results. The role of the server in this system is to create indicator definitions, distribute them to the client applications and consecutively collect and store the results.

The server consists of a web service for communication with the clients, written in PHP and an administrative web application, created with frameworks AngularJS and Bootstrap.

Keywords: indicators, compromise, IOC, server

Obsah

Úvod

V boji s malvérom je analýza a dokumentácia útokov porovnateľne dôležitá ako obrana pred nimi. Vedomosti o tom, ako bola bezpečnosť narušená, čo bolo cieľom útoku a aké škody boli spôsobené pomáhajú bezpečnostným expertom odhaliť a odstrániť slabosti a tým zvýšiť ochranu pred ďalšími útokmi.

Vyšetrovanie bezpečnostných incidentov v štátnej informačnej a komunikačnej infraštruktúre je jedna z hlavných úloh jednotky CSIRT.sk (Computer Security Incident Response Team – Jednotka pre riešenie počítačových incidentov zriadená Ministerstvom financií).

Práve CSIRT.sk je pôvodným zadávateľom tejto práce. Systém, pomocou ktorého by mohli efektívne kontrolovať prítomnosť indikátorov kompromitácie (rôznych stôp zanechaných po narušení bezpečnosti) na veľa počítačoch naraz, by im uľahčil prácu.

V prvej kapitole sú popísané indikátory kompromitácie a ako ich využitie v informačnej bezpečnosti. Druhá kapitola pojednáva o pôvodnej špecifikácii projektu. Tretia kapitola podrobnejšie popisuje funkcie serveru pre systém na detekciu indikátorov kompromitácie. Štvrtá kapitola sa zaoberá použitými technológiami a piata niektorými špecifikami implementácie.

Kapitola 1

Indikátory kompromitácie

V prípade že bola bezpečnosť počítača kompromitovaná, v systéme ostanú stopy takéhoto narušenia [10]. Tieto stopy môžu byť rôzneho charakteru, od jednoduchých metadát až po zložité kúsky škodlivého kódu. Skupine takýchto stôp, charakterizujúcich nejaké špecifické narušenie sa hovorí indikátor kompromitácie (anglicky indicator of compromise, IOC) [5].

Indikátory kompromitácie sa dajú použiť pri analýze narušení bezpečnosti. Prítomnosť známych IOC pomáha klasifikovať typ útoku [5] a nové útoky sa dajú zdokumentovať vytvorením indikátorov podľa výsledkov ich analýzy [8, kap. 1.2.].

Aby sa indikátory kompromitácie dali efektívne použiť, mali by vedeť popísať priebeh útokov na systém. Rôzne typy narušení bezpečnosti (nepovolené vniknutia, infekcie malvérom a pod.) sú však veľmi odlišné. Na popis takýchto komplexných udalostí bolo vyvinutých viacero formátov pre zápis indikátorov kompromitácie [8, kap. 3.1.]. Ako ilustráciu funkčnosti IOC uvádzame popis formátu OpenIOC.

1.1 OpenIOC

OpenIOC je formát na na popis indikátorov kompromitácie vyvinutý firmou Mandiant. Je založený na XML, takže sa dá ľahko spracovať počítačom a zároveň je pochopiteľný pre ľudí [8, kap. 1.2.]. OpenIOC umožňuje spájať termíny (jednoduché indikátory) do logických stromov, čo v kombinácii s veľkým množstvom podporovaných typov termínov a rozšíriteľnosťou XML spôsobuje jeho veľkú flexibilitu [10].

Okrem informácií o samotných indikátoroch, OpenIOC súbory obsahujú dodatočné dáta na zjednodušenie ich využitia. Súbor začína hlavičkou, popisujúcou verziu XML a OpenIOC schémy, podľa ktorej je čítaný. Nasledujú informácie o definičnom súbore, ako jeho autor, názov, kľúčové slová alebo čas a dátum vytvorenia. Za nimi nasleduje samotné telo súboru, obsahujúce popisy jednotlivých indikátorov kompromitácie. Príklad celého OpenIOC súboru je uvedený v obrázku 1.1.

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     id="6d2a1b03-b216-4cd8-9a9e-8827af6ebf93" last-modified="2011-10-28T19:28:20"
     xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Zeus</short_description>
  <description>Finds Zeus variants, twexts, sdra64, ntos</description>
  <keywords />
  <authored_by>Mandiant</authored_by>
  <authored_date>0001-01-01T00:00:00</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="9c8df971-32a8-4ede-8a3a-c5cb2c1439c6">
      <IndicatorItem id="50455b63-35bf-4efa-9f06-aeba2980f80a" condition="contains">
        <Context document="ProcessItem" search="ProcessItem/name" type="mir" />
        <Content type="string">winlogon.exe</Content>
      </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="9f7a5703-8a26-45cf-b801-1c13f0f15d40">
        <IndicatorItem id="cf77d82f-0ac9-4c81-af0b-d634f71525b5" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir" />
          <Content type="string">Mutant</Content>
        </IndicatorItem>
        <IndicatorItem id="a1250d55-cd63-46cd-9436-e1741f5f42c7" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
          <Content type="string">__SYSTEM__</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

Obr. 1.1: Príklad OpenIOC súboru. Získaný z `http://openioc.org/iocs/6d2a1b03-b216-4cd8-9a9e-8827af6ebf93.ioc`. Upravený.

Výsledné OpenIOC definície dokážu popisovať jednoduché IOC ako napríklad hľadanie súboru podľa jeho názvu, MD5 hashu, veľkosti, alebo zisťovanie prítomnosti entít v pamäti (bežiace procesy, mutexy). Využitím logických operátorov sa dajú dosiahnuť zložitejšie vyhľadávania, ako napríklad detegovanie viacerých jednoduchých podmienok naraz, alebo vyhľadávanie súborov, ktoré sa v danom priečinku nachádzať nemajú [10].

1.2 Využitie IOC

Formáty na popis IOC sami o sebe predstavujú iba definície určitých stôp zanechaných v systéme. Na to, aby sa dali prakticky využiť sú potrebné aplikácie, ktoré počítače prehľadajú a pokúsia sa nájsť indikátory zhodné s definíciami.

Indikátory kompromitácie spolu v spojení s aplikáciou na ich hľadanie sa dajú jednoducho použiť pri vyšetrovaní narušení bezpečnosti. Keď je zistené nejaké narušenie na počítačoch, bezpečnostní technici ho vyšetria a identifikujú jeho kľúčové kroky. Na základe týchto znalostí o priebehu útoku vedia vytvoriť indikátory, ktoré sa následne dajú distribuovať v rámci podniku a použiť na detekciu kompromitácie na iných počítačoch. Novo získané výsledky umožňujú doladiť detaily IOC (falošné zhody, nové vniknutia), ktoré sa dajú opätovne použiť na prehľadávanie v ostatných systémoch. Keď sú už rozsah kompromitácie a spôsobené škody známe, bezpečnostný tím môže jednoducho prejsť k nápravným opatreniam [10].

Kapitola 2

Špecifikácia projektu

Cieľom projektu je vytvoriť detekčný systém, pomocou ktorého sa bude dať kontrolovať zabezpečenie počítačov prostredníctvom indikátorov kompromitácie. Nájdenie indikátorov známych typov útokov v systéme pomôže odhaliť slabé zabezpečenie a opraviť spôsobené škody.

Požiadavky na funkčnosť výsledného systému boli formulované už pri zadaní práce bezpečnostnými technikmi z CSIRT.sk. Podľa špecifikácie sa systém má skladať z dvoch častí - klienta a servera. Pôvodná špecifikácia sa nachádza v dodatku A. V priebehu práce na projekte bola po konzultáciách špecifikácia mierne upravená.

2.1 Klient

Klientská aplikácia bude prehľadávať počítače, detegovať v nich dané indikátory kompromitácie a generovať hlásenia podľa nájdených zhôd. Medzi požadované indikátory patria prítomnosť súborov na disku, procesov v pamäti, prebiehajúca sieťová komunikácia, záznamy v DNS cashe systému, prítomnosť certifikátov v systéme alebo výskyt kľúčov v registroch Windows.

Má byť schopná bežať na rôznych verziách operačného systému Windows ako pre osobné počítače tak aj pre servery, pri čom má byť prenosná (nevyžaduje inštaláciu). Pracovať má v dvoch režimoch. Pri samostatnom režime bude získavať definície indikátorov a ukladať výsledky v rámci lokálneho adresára, zatiaľ čo pri sieťovom režime na tieto činnosti využije server.

2.2 Server

Serverová aplikácia poskytne klientom definície indikátorov kompromitácie a následne prijme výsledné hlásenia. Taktiež umožní spravovať systém cez webové rozhranie - prezerať, upravovať, prípadne importovať a exportovať definičné súbory IOC. Výsledky

kontrol získané od klientov sa budú dať prehľadávať a filtrovať podľa viacerých kritérií (zariadenie, na ktorom kontrola prebehla, dátum kontroly, nájdené indikátory).

Komunikácia medzi klientom a serverom má byť zabezpečená. Spojenie bude prebiehať prostredníctvom protokolu HTTPS a klient sa so serverom majú navzájom autentifikovať na základe bezpečnostných certifikátov.

Funkčne bude server pozostávať z 3 častí:

- *Webovú aplikáciu* na správu systému
- *Webovú službu*, cez ktorú bude klient schopný komunikovať so serverom
- *Databázu*, v ktorej budú uložené informácie o IOC a výsledky kontrol

Kapitola 3

Server

V tejto kapitole podrobnejšie popíšeme plánovanú funkčnosť servera a detaily komunikácie s klientmi. Najprv objasníme ako fungujú webové API (application programming interface) založené na návrhovom vzore Model-View-Controller a akú úlohu zohrajú vo výslednom fungovaní serveru. Potom popíšeme formát, pomocou ktorého bude server komunikovať s klientmi a na záver funkcie webovej aplikácie na správu.

3.1 Model-View-Controller

Model-View-Controller (MVC) je jeden z mnohých návrhových vzorov využívaných pri dizajne aplikácií. Jeho podstatou je rozdelenie funkcií aplikácie na 3 časti:

- *Model* - model reprezentujúci dáta, s ktorými aplikácia pracuje
- *View* - náhľad, ktorým sú používateľovi dáta prezentované
- *Controller* - ovládač, pomocou ktorého používateľ interaguje s dátami

Ako príklad vzoru MVC si môžeme predstaviť hru, v ktorej sa hráč snaží nájsť cestu bludiskom zo živého plotu. Ako model by slúžil plán bludiska, určujúci kadiaľ sa hráč môže pohybovať, alebo kde začína. View by bolo to, čo hráč vidí. V jednom variante hry sa bude pozerať na bludisko z prvej osoby, ako keby v ňom stál. V inom môže mať pohľad z vtáčej perspektívy, takže namiesto toho aby videl iba chodbu pred sebou vidí celé bludisko. Conroller by bola sada povolených akcií, ktoré môže hráč vykonať. Obyčajný návštevník sa môže bludiskom iba pohybovať, zatiaľ čo záhradník má k dispozícii aj nožnice, ktorými vie plot ostrihať a skrátiť si cestu.

Vzor MVC, vychádzajúci z tvorby používateľských rozhraní [4] sa spája najmä s grafickými aplikáciami. Vďaka tomu, že dáta sú oddelené od grafickej časti, výzor aplikácie sa dá jednoducho zmeniť a bez nutnosti meniť štruktúry v pozadí. Taktiež sa dajú ľahko vytvoriť viaceré varianty zodpovedajúce rôznym používateľským oprávneniam, keďže môžu mať samostatné ovládače [7]. Vďaka faktu, že MVC delí aplikáciu na

vrstvy umožňuje využitie tohto návrhového vzoru aj vo webových aplikáciách. Niektoré časti aplikácie môžu bežať na serveri, zatiaľ čo iné u klienta [7]. Pri takto štrukturovanej aplikácii môže byť väčšie množstvo klientov (napr. mobilná aplikácia, PC aplikácia, webstránka), pri čom všetci pracujú s tým istým serverom.

3.2 Web API

API je všeobecný pojem označujúci sadu metód, ktoré sa používajú na tvorbu aplikácií. Môžu to napríklad byť vstavané funkcie programovacieho jazyka, alebo funkcie nejakej použitej knižnice.

V oblasti webových aplikácií API nadobúda trochu iný význam. Práca s webovým API väčšinou prebieha pomocou HTTP spojení. Namiesto volania funkcií pri štandardnom programovaní sa klient pripojí k serveru a odošle mu požiadavku. Server túto požiadavku vyhodnotí a odpovie klientovi s výsledkom.

API v takejto forme priamo zodpovedá MVC architektúre. Formát požiadaviek a to, aké funkcie sú takto sprístupnené tvoria controller. Funkcie vytvárajúce odpoveď sú view a dáta použité pri jej pripravovaní sú model.

V prípade nášho servera sa model stará o komunikáciu s databázou - získavať z nej dáta, ktoré si klient vyžiada, prípadne v nej upravovať záznamy podľa zmien urobených cez webové rozhranie. Klient so serverom interaguje pomocou jednej sady funkcií, zatiaľ čo administrácia má k dispozícii iné. Výsledky volaní sú pre obe služby prezentované rovnako (odpoveď servera). Ich ďalšou interpretáciou sa API nezaoberá.

Komunikácia s API prebieha cez HTTPS pomocou metódy POST - parametre HTTP požiadavky obsahujú detaily volania API funkcie. Server z požiadavky zistí, akú funkciu sa chce klient zavolať a pokúsi sa ju vykonať. Následne klientovi vráti výsledky ako odpoveď na jeho POST požiadavku.

3.2.1 Volania

Keďže API v sebe obsahuje viacero controllerov s rôznymi funkciami, volania musia obsahovať nie len parametre funkcie, ale aj popis toho, ktorá funkcia sa má zavolať. Celkovo, POST požiadavky majú nasledovné parametre:

- `controller` - meno controllera, ktorý má byť použitý
- `action` - žiadaná akcia (meno funkcie)
- dodatočné pomenované parametre, špecifické pre volanú funkciu

3.2.2 Funkcie

Pre klientskú aplikáciu sú dostupné iba 2 základné funkcie: získanie definícií pomenovanej sady indikátorov zo servera a nahratie výsledkov naspäť na server.

Správcovská časť API obsahuje viacero skupín funkcií. Funkcie na pridávanie a upravovanie základných indikátorov kompromitácie, funkcie na vytváranie definičných súborov a funkcie na prezeranie výsledkov kontrol.

V rámci skutočnej štruktúry API sú tieto skupiny funkcií reprezentované samostatnými controllermi. Aby nedošlo k neželaným zmenám v databáze, volanie administratívnych controllerov vyžaduje špeciálne práva.

3.2.3 Odpovede

Odpovede servera neobsahujú iba samotné výsledky API volaní. V odpovedi sa vždy nachádza stavová hodnota, hovoriaca o úspechu volania. Ak počas vyhodnocovania alebo výpočtu volania nastala chyba, odpoveď tiež obsahuje chybovú hlášku popisujúcu problém.

Aby sa dali odpovede jednoducho spracovať klientmi aj webovou aplikáciou, sú odosielané vo formáte JSON.

JSON

JSON (JavaScript Object Notation) je textový formát, slúžiaci na reprezentáciu dát. Jeho štruktúra je založená na syntaxi JavaScriptu, ale sám o sebe je jazykovo nezávislý.

JSON umožňuje reprezentovať základné dátové typy (čísla, reťazce, pravdivostné hodnoty) ako aj prázdne hodnoty (null), polia (usporiadané zoznamy) a slovníky (súbory pomenovaných hodnôt). Prvky polí a hodnoty v slovníkoch tiež môžu byť ľubovoľným z týchto typov, vďaka čomu sa JSON dá použiť na reprezentáciu takmer ľubovoľnej dátovej štruktúry.

Celkovú štruktúru formátu JSON ako aj reprezentáciu rôznych typov dát možno vidieť na obrázkoch 3.1 a 3.2.

3.2.4 Formát dát

Keďže celá komunikácia medzi klientmi a serverom pozostáva zo sťahovania definícií a nahrávania výsledkov, navrhli sme pre tieto dáta štruktúry, pomocou ktorých sa dajú efektívne reprezentovať.

Definície IOC

Základom definičného súboru je zoznam indikátorov, ktoré majú byť overené. Tieto indikátory môžu byť jednoduché (názov súboru a pod.), alebo to môžu byť logické

```
[
    {
        "id": 5,
        "name": "Mutex",
        "type": "mutex-name",
        "value": [
            "semaphore"
        ]
    },
    {
        "id": 3,
        "name": "#3",
        "type": "and",
        "children": [
            {
                "id": 6,
                "name": "FMFI UK",
                "type": "dns",
                "value": [
                    "fmph.uniba.sk"
                ]
            },
            {
                "id": 2,
                "name": "Explorer process",
                "type": "process-name",
                "value": [
                    "explorer.exe"
                ]
            }
        ]
    }
]
```

Obr. 3.1: Príklad definície IOC. V takomto formáte získava klient definície zo servera.

spojky. Logické spojky v sebe obsahujú zoznam ďalších IOC, ktoré sú ich potomkami v logickom strome. Pre týchto potomkov platia tie isté pravidlá, ako pre indikátory v základnom zozname definičného súboru (jednoduché IOC a ďalšie logické operátory).

Jednoduché indikátory v sebe obsahujú názov, typ, hodnoty a identifikačné číslo (ID). Rôzne typy IOC môžu mať rôzne počty hodnôt – napr. meno certifikačnej autority má jednu hodnotu, hash procesu má dve (typ a hodnota hashu). ID sa používa na rozlíšenie indikátorov vo výsledkoch kontroly.

Príklad definície IOC vo formáte JSON je v obrázku 3.1.

Výsledky

Výsledkový súbor začína hlavičkou, ktorá obsahuje údaje o kontrole. Tie sa skladajú z názvu organizácie, mena počítača, času kontroly a mena použitého definičného súboru. Tieto informácie slúžia na to, aby technici vedeli, v ktorom klientovi boli indikátory

```
{
    "org": "Organizacia",
    "dev": "POCITAC1",
    "timestamp": 1461740751,
    "set": "Quick scan",
    "results": [
        {
            "id": 5,
            "result": 0,
            "data": []
        },
        {
            "id": 6,
            "result": 1,
            "data": ["C:\\folder\\", "file.txt"]
        }
    ]
}
```

Obr. 3.2: Príklad výsledkov kontroly. Server očakáva výsledky od klienta v tomto formáte.

nájdené.

Za hlavičkou nasledujú samotné výsledky – zoznam záznamov, pre každý indikátor z definičného súboru. Záznamy obsahujú ID indikátoru, či bola zistená jeho prítomnosť a prípadne doplnkové výsledky, podľa typu indikátoru.

Pri nahrávaní výsledkov na server jeden z parametrov API volania obsahuje reťazec s týmito dátami vo formáte JSON (príklad v obrázku 3.2)

3.3 Webová aplikácia

Webová aplikácia správcom systému umožňuje pripravovať definície indikátorov kompromitácie a prehľadávať výsledky získané od klientov. Taktiež poskytuje možnosť importovať a exportovať dáta na serveri vo viacerých formátoch. Všetky funkcie sú realizované prostredníctvom vyššie popísaného API.

3.3.1 Indikátory

Jednoduché IOC sú zobrazené v zozname, ktorý sa dá prehľadávať, aby sa uľahčila práca s väčším množstvom dát. Pre každý indikátor je možné zobraziť jeho podrobnosti alebo ich upraviť. Taktiež je možné indikátory pridávať, mazať alebo obnovovať tie, ktoré boli vymazané.

3.3.2 Definičné súbory

Na prácu s definičnými súbormi je dostupný nástroj na vytváranie a upravovanie ich stromové štruktúry. Vrcholom, ktoré majú potomkov (koreň stromu, logické operátory), je možné pridávať alebo odoberať synov. Jednoduché indikátory sa dajú prezerať, upravovať a tiež je možné vytvoriť úplne nové indikátory.

3.3.3 Výsledky kontrol

Zoznam výsledkov sa dá filtrovať podľa viacerých kritérií:

- meno organizácie
- meno počítača
- čas a dátum kontroly (prípadne rozsah od - do)
- názov definičného súboru
- názov samotného indikátoru
- prítomnosť indikátoru

Jednotlivé filtre sa dajú kombinovať, takže sa aj pri veľkom počte kontrol dajú rýchlo nájsť hľadané výsledky.

3.3.4 Import a export

Aby sa dali dáta zálohovať a opätovne obnovovať, indikátory, definície aj výsledky sa dajú exportovať aj importovať vo formáte JSON.

Indikátory sú reprezentované jednoduchým zoznamom, obsahujúcim ich názvy, typy a hodnoty. Okrem formátu JSON sa indikátory dajú exportovať aj vo formáte CSV (comma separated values – hodnoty oddelené čiarkou), ktorým sa dajú reprezentovať jednoduché tabuľky dát. IOC v tomto formáte sa potom dajú prezerať a upravovať v externých programoch.

Formát na zálohu definícií je založený na tom, ktorý sa používa na komunikáciu s klientmi (zoznam pomenovaných definícií). Z exportovaného súboru sa dá jednoducho skopírovať požadovaná definícia, ktorá sa potom dá použiť klientmi v samostatnom móde.

Keďže definičný súbor obsahuje aj všetky informácie o indikátoroch, ktoré sa v ňom nachádzajú, dá sa importovať aj do databázy, ktorá tieto indikátory neobsahuje. Pri importe sa najprv vytvoria všetky potrebné IOC, a potom definičný súbor, ktorý ich používa.

Tak ako definície, aj výsledky používajú na export formát kompatibilný s formátom použitým klientom. Zatiaľ čo pri definíciách bola táto vlastnosť dôležitejšia možnosť exporte, pri výsledkoch je podstatnejší import. Záznamy o kontrole vytvorené klientmi v offline režime sa dajú skopírovať a importovať do servera týmto spôsobom.

Okrem importu vo formáte JSON je možný aj export, ale tento nie je až tak efektívny ako export definícií. Pri importe výsledkov do inej databázy nie je isté, že ID indikátorov vo výsledkoch zodpovedajú indikátorom v databáze. Exportované výsledky sa stále dajú spracovávať v externých programoch, ale na to je vhodnejší formát CSV ako JSON. Výsledky vo formáte CSV sú ešte upravené, aby bola práca s nimi jednoduchšia – namiesto ID indikátorov obsahujú ich detaily – ale import z tohto formátu nie je podporovaný.

Kapitola 4

Technológie

Server pre systém na detekciu indikátorov kompromitácie (ďalej IOC server) sa skladať z viacerých častí (HTTP server, API, databáza a webová aplikácia). Každá z týchto častí má špecifické funkcie, a tým pádom sú na ich implementácii vhodné rozličné technológie. V tejto kapitole podrobnejšie popíšeme technológie použité pri vývoji IOC serveru, čo umožňujú a prečo sme ich zvolili.

4.1 Vývojové prostredie

Na vývoj a testovanie serveru sme použili prostredie XAMPP. Je to voľne dostupný, multiplatformový serverový balík, obsahujúci distribúcie Apache HTTP serveru, databázy MariaDB a interpreterov pre jazyky PHP a Perl (z iniciál ich mien je vytvorená skratka XAMPP).

S prostredím XAMPP sa jednoducho pracuje, a obsahuje v sebe všetky technológie, ktoré IOC server používa. Vďaka tomu sa dá projekt jednoducho vyvíjať a testovať.

4.2 Webový server

Všetky požiadavky na serveru prebiehajú cez HTTP protokol (presnejšie, cez HTTPS). Požiadavky môžu napríklad byť od klienta, ktorý chce zo servera stiahnuťi definície IOC alebo od administrátora, pristupujúceho k webovej aplikácii. Úlohou webového serveru je tieto požiadavky spracovávať a odpovedať na ne.

Pre IOC server sme sa rozhodli použiť Apache HTTP Server. Apache, ktorý je v súčasnosti najpoužívanejší web server na svete [9]. Zvolili sme ho pre to, že s ním máme skúsenosti, dá sa jednoducho nastaviť a spĺňa všetky požiadavky položené v špecifikácii.

Jednou z hlavných kvalít Apache web serveru je jeho rozšíriteľnosť pomocou doplnkových modulov. Na komunikáciu cez HTTPS slúži modul `mod_ssl`. Tento modul sa stará o naväzovanie HTTPS komunikácie, a tiež poskytuje kryptografické funkcie,

ktoré sa pri komunikácii používajú [3].

HTTPS

HTTPS funguje ako štandardný HTTP protokol, iba s rozdielom, že komunikácia prebieha po zabezpečenom sieťovom spojení. Zabezpečenie, realizované protokolom TLS (transport layer security), poskytuje viaceré výhody.

V prvom rade je komunikácia šifrovaná, takže dáta sú chránené pred čítaním a zmenami tretími stranami. Použitie obojstrannej autentifikácie bezpečnostnými certifikátmi taktiež chráni server pred nepovolenými prístupmi, keďže bez správneho certifikátu server odmietne nadviazať spojenie. Ďalšou výhodou použitia klientských certifikátov je možnosť prideliť každému klientovi iný certifikát. Tie sa potom dajú použiť na identifkáciu jednotlivých klientov a môžu nahradiť tradičné prihlasovanie menom a heslom.

4.3 API

Webové API tvorí základ funkčnosti celého serveru. Pre klientov slúži ako prístupový bod na komunikáciu so serverom a webová aplikácia ho využíva na spravovanie databázy. Táto časť je naprogramovaná v jazyku PHP.

PHP

PHP (rekurzívna skratka pre PHP: hypertext preprocessor) je programovací jazyk, vyžívaný hlavne na programovanie serverových skriptov a dynamických web stránok. Jednou z jeho silných stránok je veľké množstvo vstavaných funkcií na vykonávanie serverových činností (práca s databázami, získavanie údajov o HTTP spojení, nastavovanie HTTP hlavičiek odpovede a pod.).

O vykonávanie PHP skriptov sa stará web server. Keď dostane požiadavku, ktorej cieľom je nejaký PHP skript, vykoná ho pomocou interpreteru a jeho výsledky vráti ako odpoveď. Pre každú požiadavku je skript spustený samostatne, takže nenastávajú problémy pri prístupe viacerých klientov naraz.

4.4 Databáza

Databáza slúži na ukladanie všetkých informácií, s ktorými IOC server pracuje. Skladá sa z viacerých tabuliek, každej určenej na iný typ dát (indikátory, typy indikátorov, definície a výsledky). IOC server používa databázu MariaDB, ktorá je založená na

známej databáze MySQL a je s ňou funkčne kompatibilná. Práca s databázou prebieha prostredníctvom jazyka SQL (Structured Query Language).

4.5 Webová aplikácia

Webová aplikácia má za úlohu nie len prezentovať dáta, ale aj poskytovať funkcie na ich správu. Aby sa takéto množstvo informácií dalo prehľadne a efektívne zobraziť, aplikácia je naprogramovaná v JavaScriptovom frameworku AngularJS a využíva CSS framework Bootstrap.

Framework

Framework, podobne ako knižnica je sada funkcií. Zatiaľ čo knižnica poskytuje funkcie, ktoré sú volané programom, framework funguje na opačnom princípe. Framework poskytuje programu kostru, ktorá má už základnú funkčnosť definovanú. Program potom definuje detaily, ale framework rozhoduje o tom, ako ich použiť.

4.5.1 AngularJS

Angular je framework zameraný na jednoduché vytváranie webových aplikácií. Základná logika Angularu je založená na modeli MVC 3.1 – dáta (model) sú zobrazené prostredníctvom HTML stránky (view) a JavaScriptový rieši interakciu s používateľom (controller). Vďaka tejto abstrakcii sú aplikácie vytvorené v Angulare prehľadné a dajú sa jednoducho ďalej rozširovať.

Okrem tejto základnej formy Angular poskytuje ďalšie užitočné funkcie pre tvorbu webových aplikácií, ako napríklad direktívy (HTML šablóny s priloženou JavaScriptovou logikou), služby (zdieľané funkcie) a navádzanie (presmerovanie na podstránku, podľa toho, akú adresu napísal používateľ do prehliadača).

4.5.2 Bootstrap

Zatiaľ čo úlohou Angularu je zaručenie funkčnosti aplikácie, Bootstrap je framework zameraný na vizuálnu časť. Bootstrap definuje pravidlá kaskádového štýlu pre rôzne triedy HTML prvkov, ktoré sa potom dajú použiť na upravenie výzoru stránky. Takto Bootstrap umožňuje napr. formátovať text alebo jednoducho rozdeliť obsah stránky do stĺpcov.

Výhodami Bootstrapu sú, že je responzívny (stránka sa vie prispôsobiť rozlíšeniu prehliadača), príjemný na pohľad a jeho použitie je jednoduchšie ako budovanie vlastného štýlu od základov.

Kapitola 5

Implementácia

V tejto kapitole podrobnejšie popíšeme niektoré detaily implementácie.

5.1 API

API je štrukturované do 3 funkčných vrstiev:

1. hlavný skript
2. ovládače
3. funkcie na prácu s databázou

Úlohou hlavného skriptu je spracovať parametre HTTP požiadavky, vykonať požadovanú funkciu a vrátiť jej výsledky vo formáte JSON. Najprv z parametrov získa názov ovládača, ktorý má byť použitý. Ak ovládač existuje, skript si vytvorí jeho inštanciu a pokúsi sa v ňom nájsť želanú funkciu. V prípade, že všetko prebehlo správne, výsledok funkcie je odoslaný ako odpoveď. Ak ale počas ľubovoľného kroku vykonávania nastala chyba, vykonávanie je prerušené, a ako odpoveď je odoslaná chybová hláška.

Ovládače sú samostatné PHP objekty, ktoré v sebe obsahujú sadu funkcií. Tieto funkcie, ktoré sú volané hlavým skriptom, získavajú potrebné dáta z databázy prostredníctvom 3. vrstvy a spracovávajú ich do výslednej podoby.

Najnižšia vrstva API sa skladá z funkcií, ktoré realizujú SQL volania na databázu. Niektoré z funkcií nepožadujú žiadne parametre (napríklad získanie všetkých IOC z tabuľky indikátorov), ale tie, ktoré ich používajú, by mohli byť potenciálnymi zraniteľnosťami voči SQL injection útokom.

SQL injection

SQL injection (vstreknutie) je typ útoku, pri ktorom je kus škodlivého SQL kódu poslaný serveru ako dáta. V prípade, že server nemá dobre ochránené jeho volania na

```
"SELECT * FROM `table` WHERE `id` = $id;"
"SELECT * FROM `table` WHERE `id` = 0; DROP TABLE `table`;"
```

Obr. 5.1: Zraniteľný SQL príkaz v PHP.

databázu, tento kód môže byť vykonaný a spôsobiť škody [6].

Na obrázku 5.1 je príklad zraniteľného SQL príkazu v PHP. Ak by sa do príkazu v prvom riadku dostala premenná `$id` s hodnotou "`0; DROP TABLE `table`;`", tento príkaz by okrem očakávaného získania dát tabuľku vymazal.

Ako ochrana pred takýmito zraniteľnosťami slúžia v SQL pripravené príkazy (prepared statement). Tieto príkazy fungujú na takom princípe, že príkaz a dáta sú do databázy poslané oddelene. Databáza sa potom nepokúša dáta vykonať, a teda škodlivý kód vložený takýmto spôsobom nebude spustený.

5.2 Webová aplikácia

Aplikácia využíva množstvo užitočných funkcií poskytnutých frameworkmi AngularJS a Bootstrap.

5.2.1 Navádzanie

Rozdelenie aplikácie na jednotlivé komponenty (prostredia na správu IOC, definícií...) prebieha pomocou Angularového navádzania. Každý komponent má svoj samostatný HTML dokument a JavaScriptový ovládač. Podľa toho, na akú adresu sa používateľ pripojí, Angular zvolí daný komponent a tým zobrazí určitú časť aplikácie.

5.2.2 Služby

Všetky činnosti, ktoré aplikácia vykonáva sú realizované pomocou volaní na API serveru. Na jednoduchý prístup k týmto volaniam sú vytvorené Angularové služby.

Služba je väčšinou nejaká funkcia, ktorá sa dá využívať v iných častiach aplikácie. Keďže API má v sebe viacero ovládačov, ktoré chceme používať, rozhodli sme sa vytvoriť služby pre každý z nich. Tým pádom naša služba nie je funkcia, ale objekt, ktorý obsahuje funkcie. Volanie API a spracovanie výsledkov je vykonávané vnútri služby, takže pri jej použití v programe ju stačí jednoducho zavolať.

5.2.3 Asynchrónne vykonávanie

Práca s API nemusí prebiehať okamžite. V prípade, že je server nedokáže požiadavku okamžite spracovať, čakanie na odpoveď môže trvať nejakú dobu. Z tohto dôvodu An-

gular vykonáva HTTP volania asynchrónne.

Pri normálnom sekvenčnom vykonávaní by aplikácia poslala serveru požiadavku a potom by čakala na odpoveď. Počas tohto čakania by sa program nevykonával, a aplikácia by sa javila zamrznutá.

Asynchrónne volania fungujú iným spôsobom: pri odosielaní požiadavky je definovaná funkcia, ktorá má spracovať výsledky, keď budú dostupné. Program na takéto volanie nečaká a pracuje ďalej. Dôsledkom tohto prístupu je, že aj všetky ďalšie operácie s týmito dátami musia byť robené asynchrónne, keďže iba v tejto asynchrónnej postupnosti funkcií si môžeme byť istý, že výsledky už existujú.

Pri Angularovej aplikácii toto našťastie nie je až aký veľký problém. Angular zmeny v modeli (dátach dostupných v aj v ovládači aj v HTML dokumente) sleduje, a ich zobrazenie v HTML upravuje v reálnom čase. Síce sa údaje zobrazia až chvíľu po tom, ako je odoslané API volanie, ale dokým nie sú dostupné nie sú nijako zobrazené. Tým pádom nemôže nastať situácia, že by sa používateľ pokúsil pracovať s dátami, ktoré ešte neexistujú.

5.2.4 Formát a štýly

Vzhľad aplikácie je viditeľný v dodatku B.

Záver

Výsledný server dostatočne spĺňa požiadavky stanovené v špecifikácii. V niektorých oblastiach špecifikáciu presahuje (obnovovanie zmazaných indikátorov, grafická reprezentácia definičných súborov) a v iných mierne zaostáva (podpora existujúcich formátov na import a export).

Nedostatky boli prekonzultované so zadávateľmi práce, a dospeli sme k rozhodnutiu, že server v tomto stave je ako výsledok bakalárskej práce postačujúci na beh v praxi. Podporuje všetky potrebné druhy IOC, vytváranie definičných súborov je jednoduché a umožňuje vytvárať aj zložité logické stromy a filtrovanie výsledkov umožňuje rýchlu orientáciu aj vo veľkom množstve dát. Importné a exportné formáty sú postačujúce na zálohovanie a kopírovanie dát medzi servermi. Tiež sa dajú použiť na manuálnu distribúciu definícií ku klientom bez prístupu na internet. To ale neznamená, že ďalšie zlepšenia nie sú možné.

Okrem implementovania chýbajúcich funkcií sa dá server zlepšovať nad rámec špecifikácie aj v ďalších oblastiach. Medzi podporované formáty na import a export by sa dali pridať nie len štandardné formáty ako OpenIOC, ale mohlo by sa vytvoriť aj rozhranie na vytváranie a úpravu vlastných formátov založených na CSV. Takto definované formáty by slúžili ako slovníky schopné prekladať dáta pri importe a exporte.

Ďalším možným rozšírením do budúcnosti by bolo priraďovanie univerzálne unikátnych identifikátorov k jednotlivým IOC. Pomocou týchto UUID sa dali rozpoznať indikátory aj po importe do iných databáz, kde by ich lokálne ID bolo iné. V spojení s funkčnosťou na vytváranie nových typov indikátorov by sa takto dala dosiahnuť veľmi efektívna rozšíriteľnosť systému.

Aj po skončení bakalárskej práce ostávame v kontakte s technikmi z CSIRT.sk, aby sme celý systém na detekciu indikátorov kompromitácie (ako server tak aj funkčnosť klientov) mohli vylepšiť na toľko, že sa stane naozaj účinným nástrojom na odhaľovanie bezpečnostných narušení.

Dodatok A

Pôvodná špecifikácia

V tomto dodatku uvádzame pôvodnú špecifikáciu poskytnutú zadávateľmi z CSIRT.sk.

Špecifikácia nástroja IOC checker

IOC checker je nástroj, ktorý na základe prítomnosti indikátorov kompromitácie v systéme identifikuje kompromitované systémy.

Všeobecný popis aplikácie:

Funkčné požiadavky

1. Aplikácia musí byť schopná detegovať nasledujúce základné indikátory kompromitácie

    - Prítomnosť súborov na základe ich názvu (aj regulárneho výrazu) a hashovej hodnoty (MD5, SHA2-256)
    - Prítomnosť procesu v pamäti na základe jeho názvu (aj regulárneho výrazu), hashovej hodnoty spustiteľného súboru (MD5, SHA2-256)
    - Prítomnosť Mutexu (Mutanta) na základe jeho názvu
    - Sieťovú komunikáciu na základe komunikujúcej IP adresy, alebo doménového mena (aj regulárneho výrazu)
    - Prítomnosť DNS záznamu v DNS cache systému
    - Prítomnosť nainštalovaného Certifikátu v systémovom úložisku certifikátov na základe:
        - Doménového mena
        - Certifikačnej autority
    - Prítomnosť kľúča v registroch Windows na základe
        - Mena kľúča (aj regulárneho výrazu)

- Cesty, Mena a hodnoty klúča

2. Aplikácia musí byť schopná detegovať zložené indikátory kompromitácie nasledujúceho typu:
   (a) A , B sú IOC $\Rightarrow$ (A $\wedge$ B) je IOC
   (b) A, B sú IOC $\Rightarrow$ (A $\vee$ B) je IOC
3. Aplikácia musí podporovať nasledujúce systémy:
   (a) MS Windows XP – Windows 10
   (b) Windows Server 2003 – 2012 R2
4. Aplikácia musí byť funkčná v dvoch režimoch
   (a) Standalone
   (b) Sieťová verzia
5. Aplikácia musí byť prenosná (portable), t.j. nevyžaduje žiadnu inštaláciu a počas svojho behu nevykonáva žiadny zápis do registrov ani na disk inde ako do vlastného adresára
6. Aplikácia v prípade standalone musí:
   (a) Čítať súbor s definíciou IOC z lokálneho adresára
   (b) Čítať súbor s konfiguráciou z lokálneho adresára
      i. Konfigurovateľné parametre:
         A. IP adresa servera
         B. Režim behu (sieťový alebo standalone)
   (c) Zapísať prítomnosť všetkých detegovaných IOC do súboru v lokálnom adresári s názvom output_%TIMESTAMP%.log
7. Aplikácia v prípade sieťového režimu musí:
   (a) Pozostávať z častí klient a server
   (b) Časť klient musí byť v rovnakom spustiteľnom súbore ako standalone režim
   (c) V časti klient podporovať nasledujúcu funkcionalitu:
      i. Čítanie definície IOC zo servera prostredníctvom HTTPS
      ii. Zapísanie prítomnosti všetkých detegovaných IOC do databázy na strane servera s jednoznačnou identifikáciou zariadenia, kde boli IOC nájdené a lokálneho aj serverového času, kedy boli IOC nájdené

iii. Čítať súbor s konfiguráciou z lokálneho adresára

iv. Konfigurovateľné parametre:

A. IP adresa servera

B. Režim behu (sieťový alebo standalone)

(d) V časti server podporovať nasledujúcu funkcionalitu:

i. Prezeranie IOC prostredníctvom webového rozhrania

ii. Autentifikáciu používateľa heslom a certifikátom

iii. Vkladanie IOC prostredníctvom webového rozhrania a importovanie IOC z definičného súboru

iv. Export a import definičného súboru do databázy prostredníctvom webového rozhrania

v. Prezeranie výsledkov kontroly IOC prostredníctvom webového rozhrania. Výsledky musia obsahovať minimálne

A. Zariadenie, na ktorom prebehla kontrola

B. Dátum a čas kontroly

C. Zoznam detegovaných IOC z databázy

vi. Filtrovanie a vyhľadávanie výsledkov kontrol IOC minimálne na základe

A. Zariadení, na ktorých prebehla

B. Intervalu dátumov a časov kontroly

C. Zoznamu detegovaných IOC

D. Logických výrazov pozostávajúcich z vyššie uvedených kritérií a spojok AND, OR, NOT

Nefunkčné požiadavky

Aplikácia sa delí na:

- Klientsku časť – klient
- Serverovú časť – server

1. Klient musí byť napísaný v C++
2. Klient musí využívať iba knižnice a komponenty štandardne dostupné v systéme Windows po inštalácií OS Windows.
3. Všetka komunikácia medzi klientov a serverom musí využívať protokol TLS
4. Klient a server v sieťovom režime sa musia obojstranne autentifikovať na základe certifikátov. Neautentifikovanému klientovi nebude umožnená komunikácia.

5. Komunikácia medzi klientom a serverom musí prebiehať prostredníctvom webových služieb.
6. Server musí zvládnuť pripojenie aspoň 100 konkurentných klientov
7. Server môže byť postavený na (prípadne dohoda)
   (a) Apache, PHP a MySQL
   (b) IIS, ASP.NET a MS SQL
8. Server musí pozostávať minimálne z:
   (a) Webového rozhrania
   (b) Webovej služby na príjem komunikácie od klientov
   (c) Databázy, ktorá uchováva informácie o IOC
9. Súčasťou aplikácie musí byť dokumentácia pre používateľov, administrátorov aj vývojárov.
10. Aplikácia musí bežať v portable móde (žiadne zápisy do registrov)

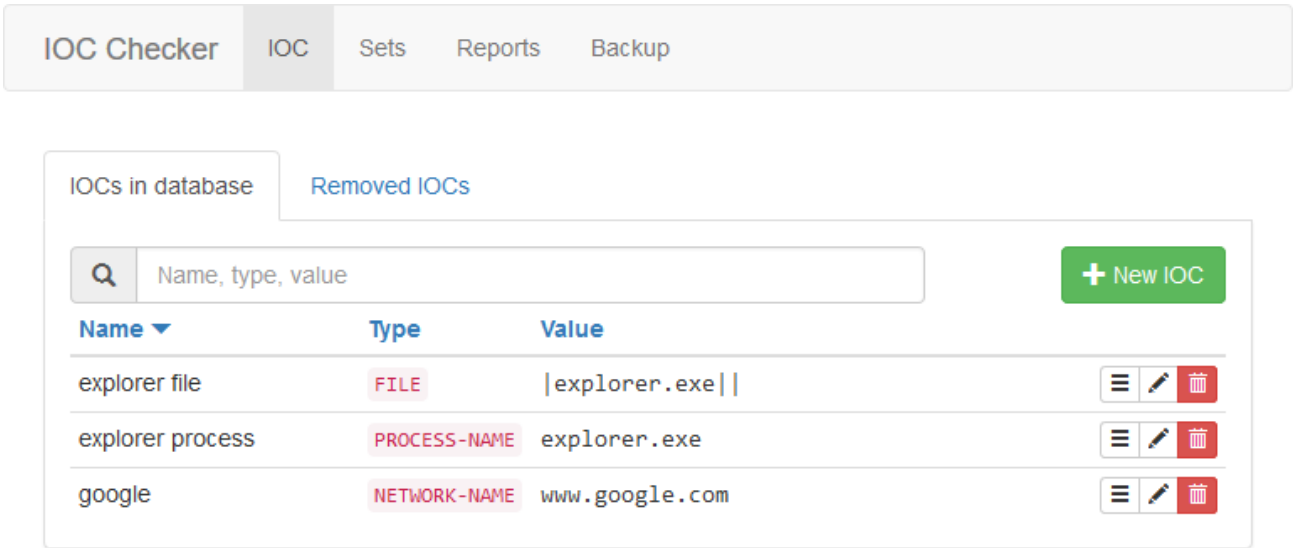
Dodatok B

Webová aplikácia

V tomto dodatku popíšeme vzhľad a funkcie webovej aplikácie.

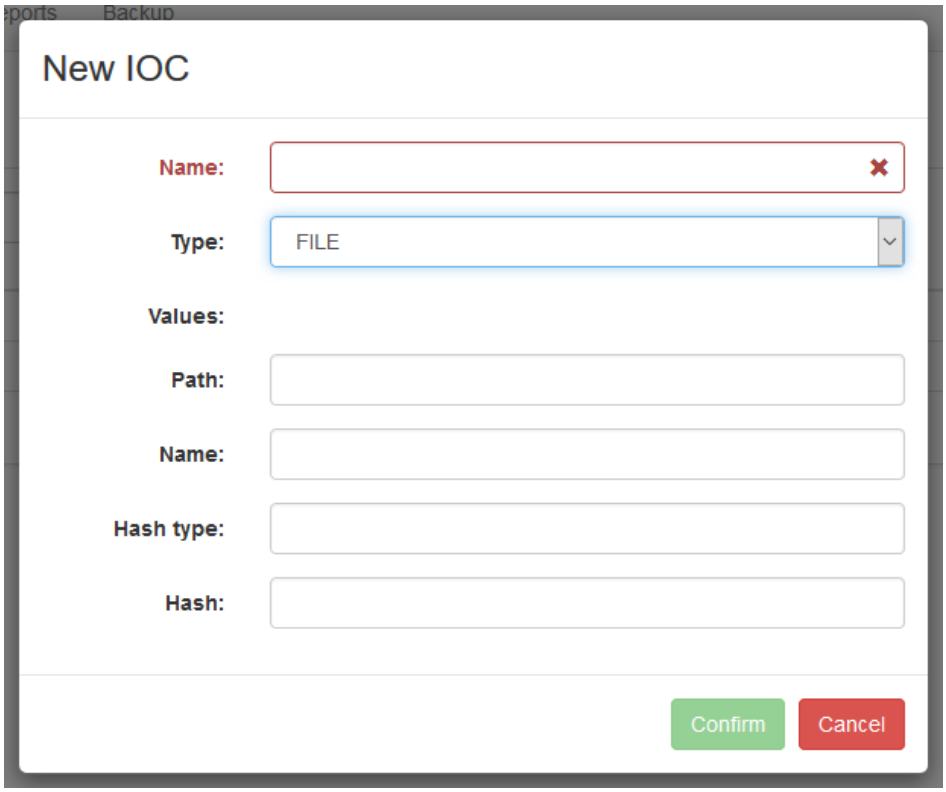
B.1 Indikátory

Základom nástroja na správu indikátorov je tabuľka (obr. B.1), ktorá obsahuje zoznam IOC. Pre každý indikátor sú dostupné ovládacie prvky – zobrazenie detailov, upravovanie a odstránenie. Tabuľku je možné filtrovať podľa hodnoty ľubovoľného z polí.



Obr. B.1: Správa IOC

Na pridávanie a upravovanie indikátorov sa používa modálny dialóg (obr. B.2). V tomto sa dá nastaviť meno indikátoru, jeho typ a hodnoty. Počet hodnôt a ich názvy závisia od typu indikátoru.



Obr. B.2: Vytváranie nového indikátoru

B.2 Definičné súbory

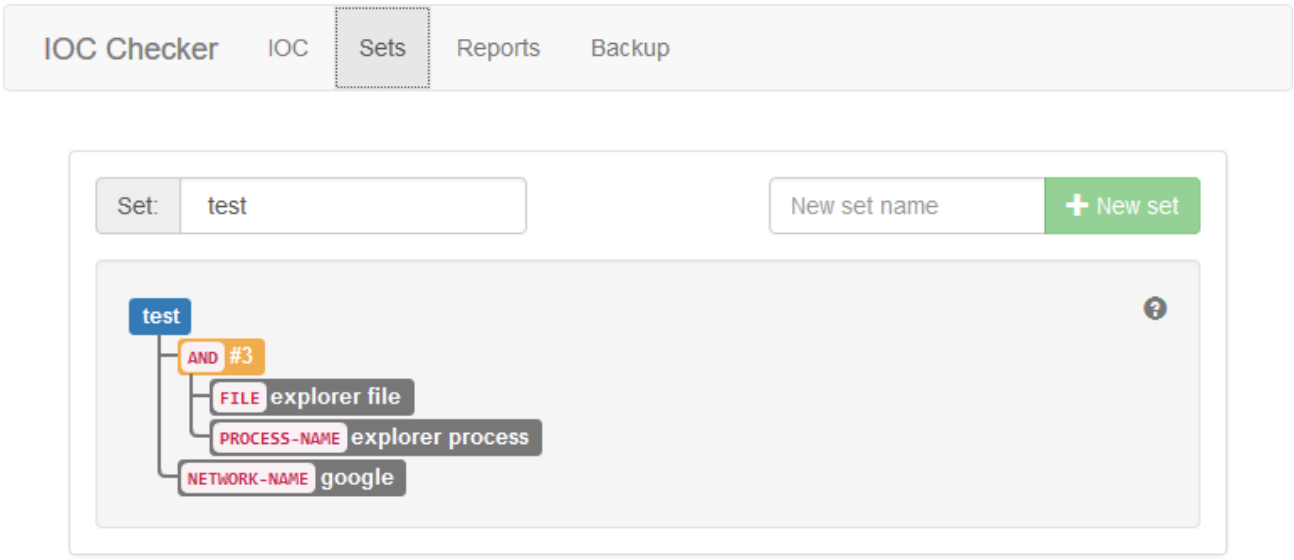
Definičné súbory sú zobrazené vo forme stromu (obr. B.3). Vyberanie, ktorý definičný súbor má byť zobrazený je riešené cez rozbaľovací zoznam. Po kliknutí na položky v strome je zobrazené kontextové menu (obr. B.4) zobrazujúce možné akcie. Základné IOC sa dajú prezerať, upravovať a odstrániť zo stromu. Logickým operátorom a koreňovému vrcholu sa dajú pridávať indikátory ako potomkovia prostredníctvom dialógového okna (obr. B.5). Logické operátory majú tiež možnosť byť odstránené.

Pridávanie indikátorov do stromu a vytváranie nového používajú ten istý dialóg. V tomto okne je možné zvoliť nejaký existujúci indikátor, vytvoriť nový indikátor (pomocou okna na obr. B.2) alebo nový logický operátor.

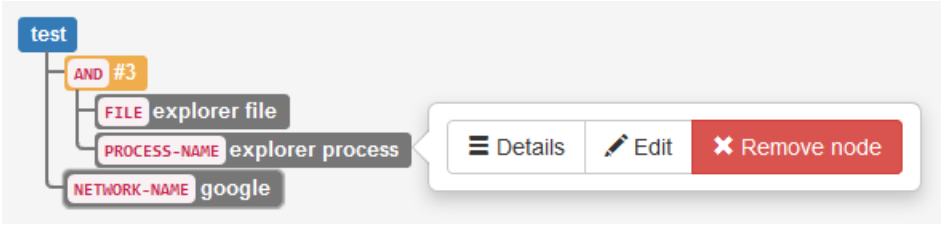
B.3 Výsledky

Na prácu s výsledkami je dostupná tabuľka, umožňujúca filtrovanie podľa každého stĺpca zvlášť (obr. B.6). Filtrovanie podľa dátumu je možné nastavením rozsahu – horná a dolná hranica sa nastavujú samostatne (obr. B.7).

Pre každý výsledok sa dajú zobraziť podrobnosti: popis indikátoru a podrobnejšie výsledky kontroly (obr. B.8).

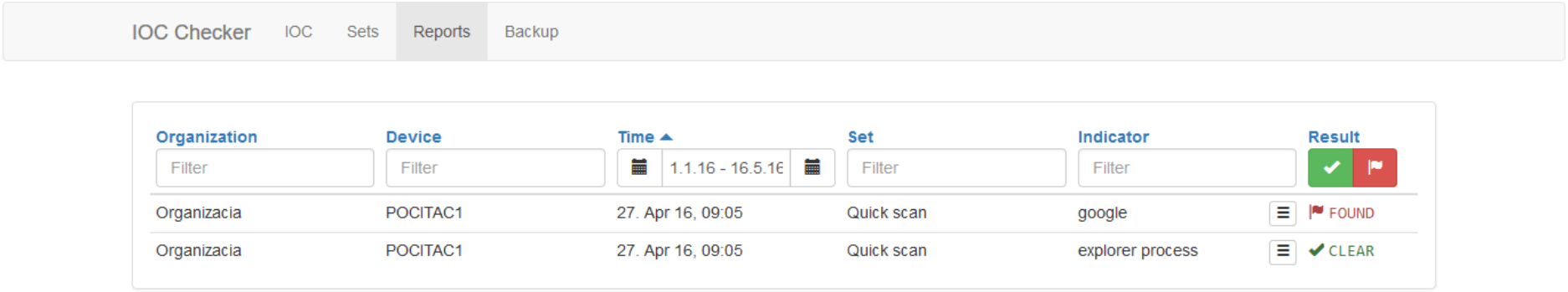


Obr. B.3: Správa definícií

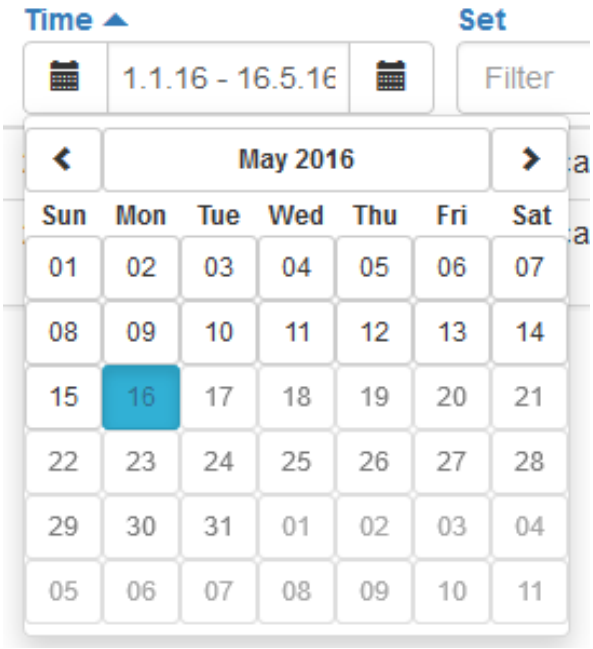


Obr. B.4: Kontextové menu

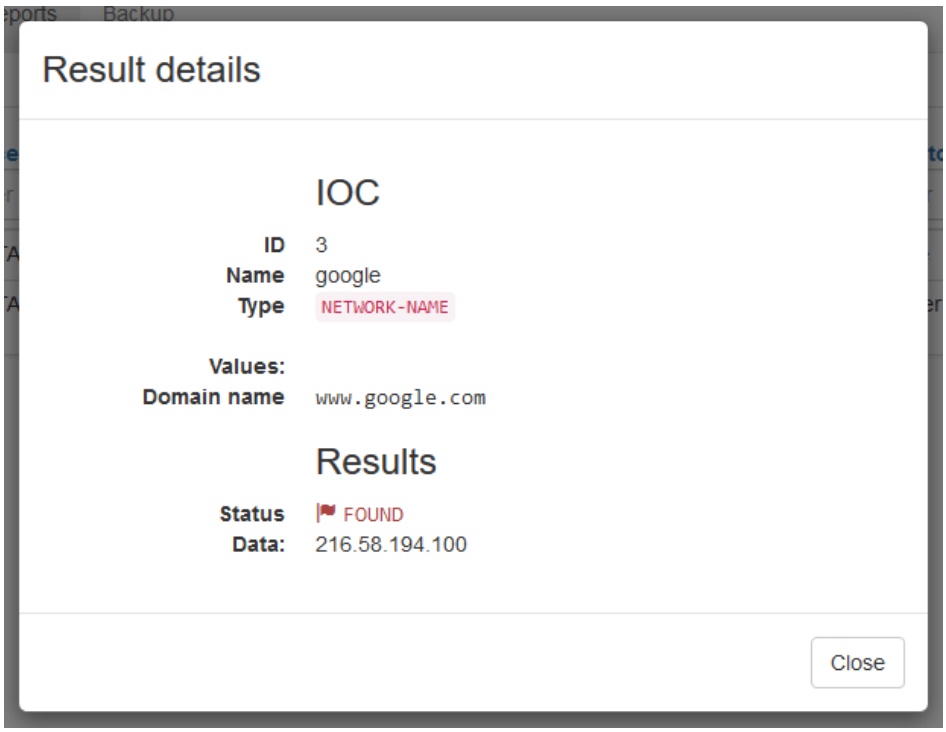
Obr. B.5: Pridávanie indikátorov do stromu



Obr. B.6: Prezeranie výsledkov



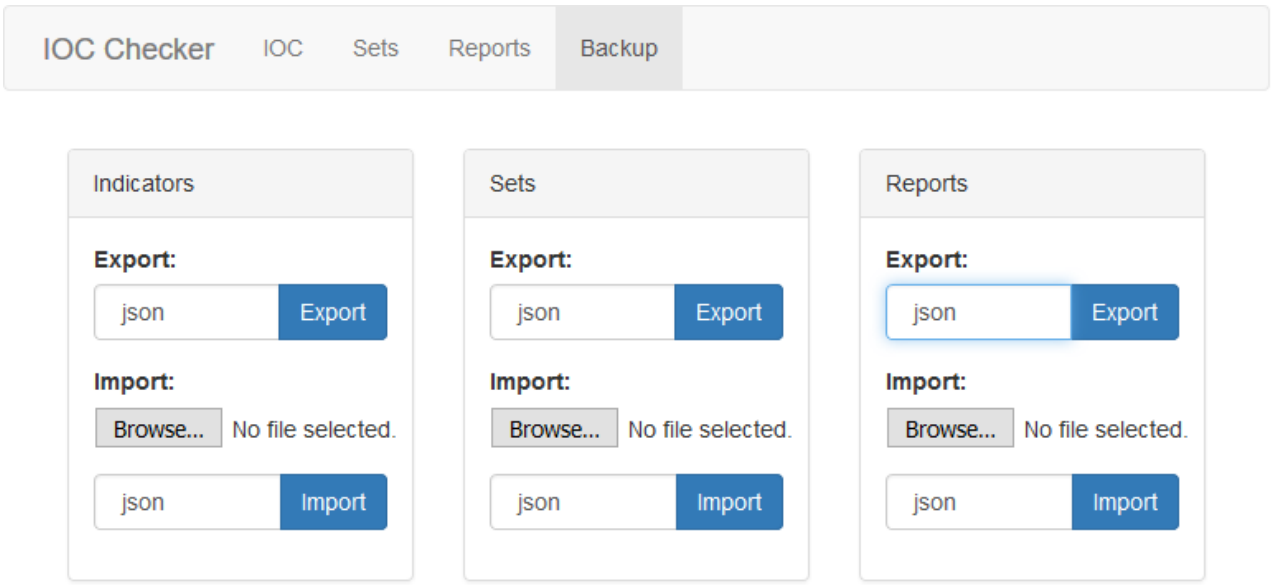
Obr. B.7: Voľba dátumového rozsahu výsledkov



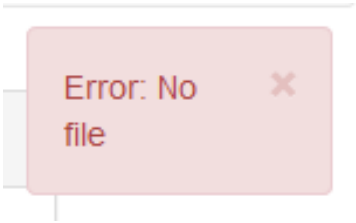
Obr. B.8: Prezeranie detailov výsledkov

B.4 Zálohovanie

Zálohovanie dát je možné prostredníctvom jednoduchého rozhrania (obr. B.9). Ak počas importu alebo exportu nastane nejaká chyba, zobrazí sa hláška obsahujúca podrobnosti (obr. B.10).



Obr. B.9: Zálohovanie



Obr. B.10: Chybové hlášky

Literatúra

[1] Callegati, F., Cerroni, W., Ramilli, M. Man-in-the-middle attack to the https protocol. *IEEE Security and Privacy*, 7(1):78–81, 2009.

[2] CSIRT.sk. FAQ. `https://www.csirt.gov.sk/faq-860.html`. [Citované 12.5.2016].

[3] Foundation, T. A. S. Apache SSL/TLS encryption. `https://httpd.apache.org/docs/2.4/ssl/`, 2015.

[4] Fowler, M. Gui architectures. `http://martinfowler.com/eaaDev/uiArchs.html`, 2006. [Citované 7.3.2016].

[5] Gragido, W. Understanding indicators of compromise (IOC) part I. `http://blogs.rsa.com/understanding-indicators-of-compromise-ioc-part-i/`, 2012. [Citované 13.1.2016].

[6] Halfond, W., Viegas, J., Orso, A. A classification of SQL-injection attacks and countermeasures. V *Proceedings of the IEEE International Symposium on Secure Software Engineering*, diel 1, 13–15. IEEE, 2006.

[7] Leff, A., Rayfield, J. T. Web-application development using the model/view/controller design pattern. V *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, 118–127. IEEE, 2001.

[8] Lock, H.-Y. Using IOC (indicators of compromise) in malware forensics. `https://www.sans.org/reading-room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200`, 2013. [Citované 13.1.2016].

[9] Netcraft. February 2016 web server survey. `http://news.netcraft.com/archives/2016/02/22/february-2016-web-server-survey.html`, 2016. [Citované 1.5.2016].

[10] OpenIOC. Sophisticated indicators for the modern threat landscape: An introduction to OpenIOC. `http://openioc.org/resources/An_Introduction_to_OpenIOC.pdf`. [Citované 20.1.2016].