Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Bakalářská práce

2017 **Luboš Beran**

Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Vulnerability management

Bakalářská práce

Luboš Beran

Školitel: Ing. Petr Břehovský

České Budějovice 2017

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Luboš Beran

Obor – zaměření studia: Kriminalisticko-technická činnost v IT

Katedra/ústav, kde bude práce vypracovávána: Ústav aplikované informatiky

Školitel: Ing. Petr Břehovský

Téma bakalářské práce: Vytvoření systému pro Vulnerability Management

Cíle práce: Cílem teoretické části práce je definovat a popsat procesy vulnerability managementu. V praktické části bude vytvořen systém, který definované procesy realizuje ve formě automatizovaných subsystémů.

Základní doporučená literatura:

[1] KANDEK, Wolfgang. *Vulnerability Management For Dummies* [online]. 2. West Sussex: John Wiley, 2015 [cit. 2016-12-03]. ISBN 978-1-119-05829-8. Dostupné z: https://www.qualys.com/docs/vm-for-dummies-2nd-edition.pdf

[2] CHUVAKIN, Anton. Vulnerability and Security Configuration Assessment Solutions Comparison. In: *Gartner* [online]. 2012 [cit. 2017-02-08]. Dostupné z: http://www.gartner.com/technology/media-products/reprints/qualys/Qualys_225382.html

[3] MELL, Peter, Tiffany BERGERON a David HENNING. SPECIAL PUBLICATION 800-40. *Creating a Patch and Vulnerability Management Program*. Version 2.0. Gaithersburg: National Institute of Standards and Technology, 2005.

Vedoucí práce: Ing. Petr Břehovský Podpis: .....

Vedoucí ÚAI: RNDr. Libor Dostálek .....

V Českých Budějovicích dne 28. 2. 2017 Podpis studenta: .....

Bibliografické údaje

Beran L., 2017: Vulnerability Management [Vulnerability Management Bc. Thesis, in Czech] - 65 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace

Tato bakalářská práce se zabývá sestavením systému pro Vulnerability Management. V teoretické části je vysvětlen pojem Vulnerability Management a popsán cyklický proces OODA. Dále jsou rozebrány rozdíly mezi penetračním testováním a Vulnerability Managementem. Na konci první části jsou uvedeny příklady databází, ze kterých nástroje pro Vulnerability Management čerpají. V praktické části uvádím příklady používaných nástrojů s jejich testováním. Jádro praktické části je v sestavení provozuschopného systému pro Vulnerability Management.

Klíčová slova

Vulnerability Management, Python, skript, PostgreSQL, databáze

Annotation

This bachelor thesis deals with creation of system for Vulnerability Management. Theoretical part contains explanation of Vulnerability Management and cyclic process OODA. Furthermore differences between penetration tests and Vulnerability Management are described here as well. Examples of databases which are used for tools within Vulnerability Management are given at the end of the theoretical part. Practical part includes testing and examples of used tools. Core of the practical part is creating the sustainable system for Vulnerability Management.

Key words

Vulnerability Management, Python, script, PostgreSQL, database

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne _____ _____

podpis

Poděkování

Rád bych poděkoval panu inženýrovi Břehovskému za odborné rady a podnětné příspěvky k práci, díky také patří mojí rodině za podporu při studiu.

Obsah

Seznam pojmů

Vulnerability Management – soubor procesů, který hledá a eliminuje zranitelnosti v systému.

Skener zranitelností – program hledající zranitelnosti na zadaném systému.

Vulnerability Assessment/systém pro posouzení zranitelností – systém, který má za úkol nalézt a ohodnotit nalezené zranitelnosti.

Zero-day vulnerability – zranitelnost, o které výrobce neví a útočníci ji dokázali využít.

GNU/GPL – licence, která dovoluje volně používat nástroje, ale musí být uveden zdrojový kód a odvozené programy musí být pod stejnou licencí [1].

MIT – licence, která dovoluje volně používat nástroje, ale nemusí být uveden zdrojový kód [1].

LGPL – licence, která je velmi podobná GNU/GPL, ale nevyžaduje, aby použité kódy byly pod stejnou licencí [1].

Backdoor – metoda napadení systému, kdy útočník obchází autentizační mechanismy a dokáže využít napadnutý PC ve svůj prospěch.

Seznam zkratek

VM – Vulnerability Management

NIST – National Institute of Standards and Technology

PVG – Patch and Vulnerability Group

OS – Operační systém

SaaS – Software as a Service

CVSS – Common Vulnerability Scoring Systém

GNU/GPL – GNU General Public License

MIT – Massachusetts Institute of Technology license

LGPL – Lesser General Public License

NIPS - Network intrusion prevention system

HIPS - Host intrusion prevention system

OODA - Observe, Orient, Decide, Act

SIEM - Security information a event management

GSA - Greenbone Security Assistant

OTP - OpenVAS Transfer Protocol

OMP - OpenVAS Manager Protocol

NVT - Network Vulnerability Tests

CLI - Command Line Interface

1 Úvod

Internet je využíván dnes a denně. Člověk si na něm hledá informace, ověřuje si je, učí se nebo vzdáleně pracuje pro zaměstnavatele. Bohužel právě zaměstnavatelé často nevěnují bezpečnosti IT pozornost. Proto by každá společnost, která pracuje s citlivými daty nebo daty o které nechce přijít, jako například s údaji o zákaznících a zaměstnancích nebo s údaji o platebních kartách, měla věnovat pozornost nejenom zabezpečení své IT struktury, ale i bezpečnosti provozovaných služeb. Odchozí zaměstnanec může použít metodu backdoor a získávat tak cenné informace, zaměstnanec může otevřít nebezpečný dokument a nakazit tak celou síť. Zkušený útočník může provést útok na citlivá data během hodiny [2].

Pro útočníka jsou zranitelnosti nejvíce zneužitelná místa, protože přes ně je schopen neoprávněného přístupu do systému. Jakmile je uvnitř, hledá důvěrné informace, čísla karet, zdravotní záznamy, firemní tajemství, zkrátka všechno, co je schopen prodat na černém trhu [3].

Správně navržený Vulnerability Management by měl, co nejvíce znesnadnit útočníkovi vniknutí do sítě tím, že bude pravidelně zjišťovat nově nalezené zranitelnosti a aplikovat záplaty. Hlavním cílem této práce je vytvořit systém, který bude ve Vulnerability Managementu využitelný. Výsledný exportovaný virtuální systém, by měl najít využitelnost v Ústavu aplikované informatiky.

1.1 Cíle práce

- Popsat Vulnerability Management a jeho procesy.
- Sestavit definici pro systém využitelný ve Vulnerability Managementu.
- Porovnat dostupné nástroje.
- Vybrané nástroje propojit do provozuschopného celku.
- Vytvořit importovatelný soubor do VirtualBoxu.

2 Vulnerability Management

Samotný Vulnerability Management, dále jen VM, prošel vývojem z prostých skriptů a programů, použitelných pouze experty, na rozsáhlé firemní frameworky, použitelné pro většinu lidí zabývajících se bezpečností se základními síťovo-bezpečnostními znalostmi [4]. Toto tvrzení podporuje i Kandek [3], který se na vývoj VM, dívá spíše ze softwarového hlediska. Zmiňuje, že VM se vyvinul z jednoduchého skeneru, na rozsáhlou aplikaci pro detekci známých slabin v systému. Popisuje také, že skenovací nástroj je stále základní element VM, ale kontinuální VM zahrnuje i další technologie a postupy. Ty přispívají především k širšímu přehledu, nezbytnému k větší kontrole a efektivnějšímu odstraňování zranitelností.

2.1 Vlastnosti Vulnerability Managementu

Podle Kandeka [3], VM znamená průběžné a systematické hledání a eliminování zranitelností v systému. VM by měl zvládat:

- udržovat databázi počítačů a zařízení připojených do sítě – hardwarová aktiva,
- sestavit seznam nainstalovaného softwaru – softwarová aktiva,
- změnit nastavení softwaru, aby byl méně přístupný napadení,
- identifikovat a opravovat chyby v instalovaném softwaru,
- upozornit na přidání nového zařízení, portu nebo softwaru do databáze a umožnit analýzu,
- umožňovat efektivní zmírnění bezpečnostních rizik,
- dokumentovat stav zabezpečení pro audity,
- nepřetržitě provádět předchozí kroky a zajišťovat tak stálou ochranu.

Chuvakin [4] zásadním stylem rozděluje nástroje určené pro Vulnerability Assessment – systém posouzení zranitelností a VM samotný. Nástroje určené pro posouzení zranitelností jsou pouze produkty, určené ke koupi, kdežto VM zahrnuje procesy, lidi a technologie.

Zdůrazňuje, že nástroje jsou kritická část, ale ne všechno. V práci je také definice, kdy nástroje pro posouzení zranitelností jsou označovány jako klíčové pro hledání a sanaci bezpečnostních chyb před tím, než jsou zneužity.

Jednoznačně pak odděluje nástroje pro VM a nástroje pro bezpečnou konfiguraci systému, kdy oba nástroje musí fungovat pospolu pro dobře fungující VM. V současné době je snaha o spojení obou do jednoho softwaru.

V normě od NIST [5] je ražen názor, že VM je úzce spojen s Patch Managementem, a to takovým způsobem, že společnost by měla vytvořit PVG, skupinu zabývající se právě těmito dvěma managementy a provádět je dohromady. Zmíněny jsou pouze povinnosti, které souvisí s VM. Vybrány byly tyto:

- vytvořit systémový inventář – zanést do seznamu všechny druhy hardwaru, softwaru a operačních systému, které se používají ve společnosti a dále manuálně udržovat databázi všech IT zdrojů nezachycených v existujícím inventáři,
- průběžně skenovat síť,
- monitorovat zranitelnosti, opravy a hrozby – monitorovat oznámení ohledně zranitelností, oprav a vznikajících hrozeb, které mají souvislost se softwarem ve společnosti,
- upřednostňování oprav zranitelností – vytvoření řádu v jakém se budou aplikovat opravy,
- ověření opravy – skenováním sítě si ověřit, že oprava byla správně instalována, dále logovat provedené změny.

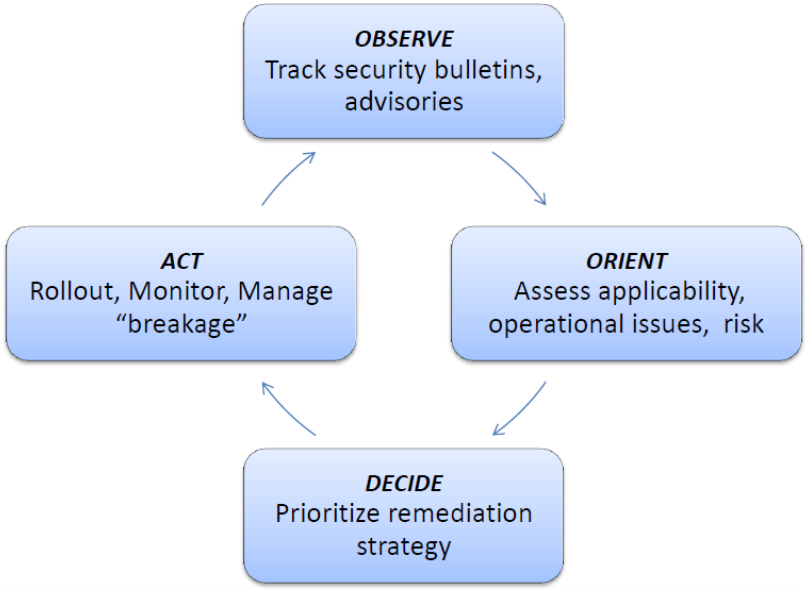
Periodické opakování procesů je důležitou součástí celého VM. Kandek [3] i Chuvakin [4] to popisují jako cyklus VM.

2.2 Cyklus Vulnerability Managementu

OODA je zkratka pro Observe, Orient, Decide a Act, přeloženo jako pozoruj, odhadni, rozhodni a jednej. Jedná se o uzavřený kruh procesů, které je nutné periodicky opakovat. Tento model byl původně vyvinut pro americkou armádu pro jednání v boji. Kandek [3]

připodobňuje zneužití zranitelností a obranu proti útočníkům jako boj armády proti nepřátelům. Model je založen na 4 jednoduchých krocích a je aplikovatelný pro práci VM.

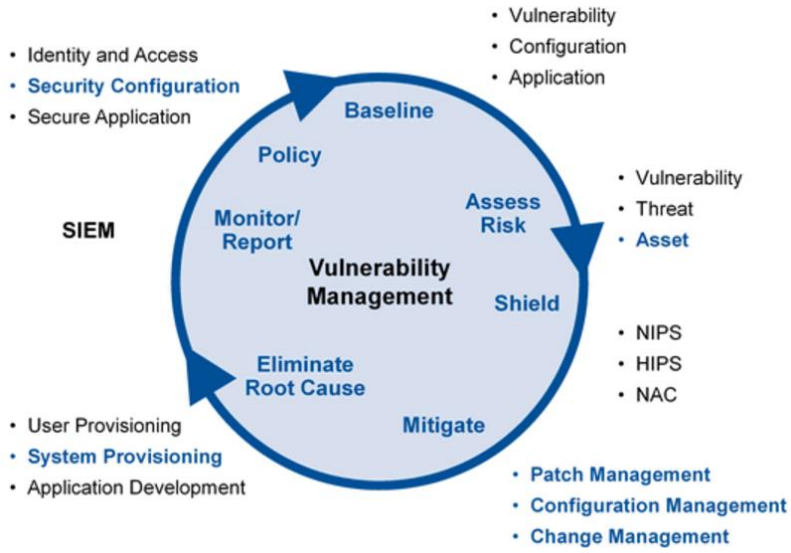
V prvním kroku Observe, by měl člověk zodpovědný za VM mít přehled o zprávách týkajících se bezpečnosti a o nově nahlášených zranitelnostech. Pro provedení druhého kroku, Orient, je nutné posoudit, jak by útočník mohl zranitelnost využít v hlídané síti. Další krok, Decide, je o rozhodnutí klíčových rizik, prozkoumat zranitelné místo a vyvinout opravnou strategii. Posledním krokem je Act, kdy je zapotřebí aplikovat záplaty a opravené místo sledovat.



Obrázek 1 OODA – převzato z [6]

Chuvakin ve své práci [4] používá velice podobný model, ale je lépe specifikováno, co se děje, ve kterém kroku.

V prvním kroku, zde baseline, má nástroj pro posouzení zranitelností za úkol skenovat síť a sbírat informace o zařízeních a zjišťovat výskyt nových zařízení. NIPS nebo HIPS dokáží dočasně ochránit systém proti zranitelnosti a Configuration Management má za úkol jí napravit. Security information a event management (SIEM) poté pomáhají s určením priority zranitelnosti a pokračujícím bezpečnostním sledováním. Pro odstranění zranitelnosti musí změna přijít v Application Development – systému pro aplikaci záplat.



Obrázek 2 Cyklus VM - převzato z [4]

Do trhu s nástroji pro posouzení zranitelností, je možno zahrnout firewally, IPS nebo SIEM.

Důležité je rozdělování povinností. Ve většině společností mají aplikaci záplat a posouzení zranitelností na starost různé týmy. Tyto týmy používají jiné nástroje pro práci. Hledání a ohodnocení zranitelností má na starosti bezpečnostní tým a aplikaci záplat tým, který se stará o server nebo o síť.

Co je důležité odnést si z tohoto článku je, že bezpečnostní tým má na starost neustálý monitoring sítě, hledání nových zařízení, hledání zranitelností a jejich hodnocení. Výsledný report, ale pak předávají jinému týmu, který se stará o aplikaci záplat. Podstatné je si uvědomit, že oba týmy musí spolupracovat.

3 Penetrační testování

V internetovém zpravodaji SecurityWeek, vyšel článek [7] o rozdílech mezi penetračním testováním (penetration testing) a posouzením zranitelností (vulnerability assessment). V článku se píše, že oba pojmy jsou často zaměňovány. Je nezbytné nejenom skenovat systém a zjišťovat zranitelnosti, ale i otestovat zranitelnost, zda je skutečně zneužitelná a jaké představuje riziko. Důležité je si uvědomit vztah mezi posouzením zranitelností a penetračním testováním.

Porovnávání zranitelností se stalo jednou z dominantních bezpečnostních praktik. Cílem této praktiky je správně identifikovat zranitelnost a dobře jí ohodnotit. Skenery jako takové jsou designované pro ohodnocení stavu bezpečnosti firmy, identifikování bezpečnostních mezer a doporučení správného postupu, buď pro eliminování nebo zmírnění slabiny na přijatelné riziko. Proces porovnávání zranitelností obvykle nalezne všechny zařízení v organizaci, které jsou klasifikovány podle hodnoty pro společnost a potencionálního dopadu. Posledním krokem je zmírnění kritických zranitelností, které by na společnost mohly mít zdrcující dopad. Čím více chyb je nalezeno, tím lépe.

Nicméně, zaměřování se na existující zranitelnosti, je pouze první krok celého procesu. Bez zohlednění využitelnosti zranitelnosti, společnosti často plýtvají svými zdroji. K lepší organizaci plánu záplat, je dobré určit, zda je zranitelnost zneužitelná. Bez zohlednění, pak společnost může řešit kritické zranitelnosti, které ovšem není možné využít a poskytuje tak útočníkovi větší prostor pro zneužívání opravdu zneužitelných chyb.

Důležité je pamatovat na to, že skenery dokáží najít chyby, které jsou nahlášeny. Jenomže pak jsou tu zero-day zranitelnosti, o kterých skener neví.

Pro větší zabezpečení ohrožených dat, čím dál více společností provádí penetrační testování. To je prováděno etickými hackery, kteří se snaží simulovat situaci, kdy je proveden útok na společnost. Cílem testování je objevit bezpečnostní trhlinu a následně zjistit jakým rizikem trhlina je a určit jaká data je schopen útočník dostat, pokud povede úspěšný útok. Výsledkem penetračního testování je zpráva o nebezpečnosti, využitelnosti a případnou opravou. Etičtí hackeři často používají automatizované nástroje jako Metasploit nebo si píší vlastní kódy.

Závěrem je důležité kombinovat popsané metody, tedy posouzení zranitelností, penetrační testování a ohodnocení rizik. Takto o rozdílech psal Torsten [7].

Druhý článek [8] vyšel v internetovém zpravodaji SecureWorks. Článek vznikl jako reakce na zaměňování pojmů posouzení zranitelností a penetračního testování. Posouzení zranitelností je proces identifikování a ohodnocení bezpečnostních zranitelností. Jde o důkladné o posouzení stavu bezpečnosti IT ve společnosti, indikování slabin a poskytnutí případných kroků ke zmírnění slabin na přijatelnou úroveň rizika.

Penetrační testování simuluje venkovní a vnitřní útok na data, kdy se útočník snaží prolomit kybernetickou ochranu. Etičtí hackeři se snaží využít chyb a získat přístup k důvěrným datům.

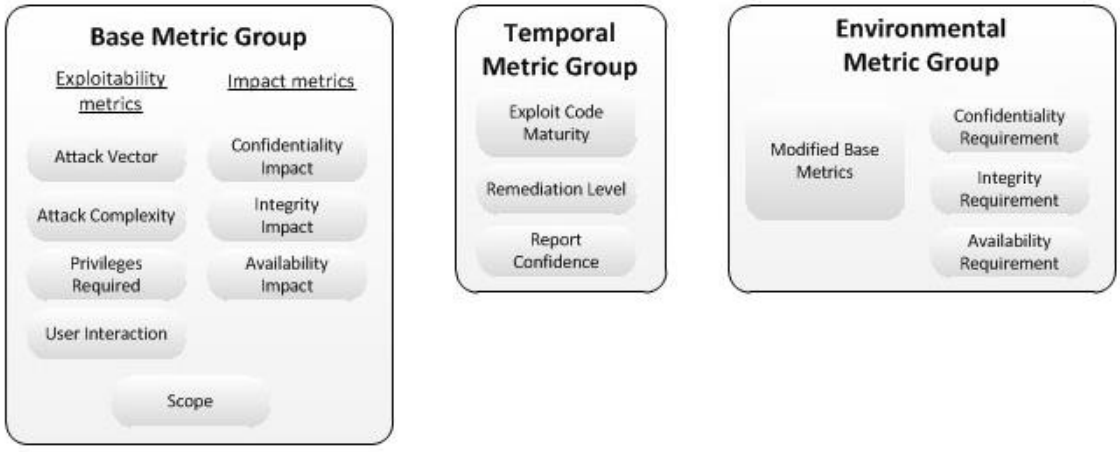
V závislosti na velikosti rozsahu penetračního testování může dojít i technikám typu sociální inženýrství. V současnosti se používají dva primární typu testů, a to black box a white box. Při typu white box hacker využívá zpráv z posouzení zranitelností, kdežto při black box testování, neví o napadaném systému nic nebo jen velmi málo.

Obě techniky, posouzení zranitelností a penetrační testování, lze určit pomocí jednoduchých otázek. Co je naší slabinou a jak jí opravit? Odpověď na tuto otázku poskytuje posouzení zranitelností. Může se někdo dostat do našeho systému a co může zjistit? Zde je odpověď penetrační testování.

Oba články se dají zkombinovat a vyvodit závěr. Společnost by měla jako první řešit bezpečnost IT pomocí posouzení zranitelností. Ustanovit plán aplikace záplat s ohledem na to, zda je zranitelnost opravdu aktuální pro danou společnost. Toto zohledňuje CVSS 3.0. Poté, pokud je společnost spokojená s bezpečností IT, je na místě provést penetrační testování a řešit případné nalezené zranitelnosti. Tedy nejprve vytvořit Vulnerability Management a až poté provést penetrační testování.

4 Common Vulnerability Scoring Systém 3.0

Common Vulnerability Scoring System (CVSS) je metrika, která se používá pro ohodnocení závažnosti nalezených zranitelností. V současné době se používá verze 3.0, k níž je dostupná i podrobná dokumentace [9]. CVSS obsahuje 3 metrické skupiny podle, které je poté, vypočítávána závažnost zranitelností. Základní skupina zohledňuje vlastnosti pojící se s napadnutým aktivem, skupina časová odráží vlastnosti, které se mění v průběhu času a poslední skupina, metriky prostředí, zohledňuje dopad zranitelnosti na společnost ze subjektivní stránky. Základní skupina dovoluje zranitelnosti nabývat hodnot 1 až 10 a ostatní 2 skupiny je mohou upravit. CVSS má především tu výhodu, že všem společnostem nabízí jednotný pohled na nalezenou zranitelnost. Společnost se také na základě výše ohodnocení může rozhodnout, zda zranitelnosti jsou nezbytně nutné opravit nebo je možné odložit jejich odstranění na později.



Obrázek 3 Základní skupiny– převzato z [9]

Základní skupina se dělí na 3 podskupiny. Exploitability metrics - metriky zneužitelnosti, Impact metrics - metriky dopadu a Scope - rozsah. V metrikách zneužitelnosti je především zohledňována komponenta, která je napadnutelná a jaké musí mít vlastnosti, aby byla napadnutelná.

Metriky zneužitelnosti se dělí na další 4 podkategorie. V Attack Vector je posuzováno, jakou cestou je možné zneužitelnost využít. Např. pokud je možné využít

zranitelnosti po internetu, dostane vyšší ohodnocení nebezpečnosti než zranitelnost, které lze dosáhnout pouze fyzickým přístupem. Attack Complexity popisuje jakou musí mít útočník kontrolu nad prostředím, pokud chce útočit. Zde jsou přípustné pouze dva stavy, nízkou a vysokou. Nízká kontrola prostředí znamená, že nejsou okolnosti, které by útočník měl zohledňovat, aby provedl úspěšný útok. Vysoká kontrola je přesný opak. Útočník tak musí zmapovat prostředí a zajistit vhodné podmínky pro zneužití. Další podkategorií je Priviliges Required. Zde se skóre zvyšuje, čím nižší je potřebné oprávnění. Ke zranitelnosti se útočník dostane hůře pokud bude potřebovat administrátorská práva. V User Interaction je skóre vyšší, pokud útočník nepotřebuje spolupráci uživatele. Pakliže je nutná interakce uživatele je skóre nižší.

Metriky dopadu obsahují 3 podkategorie a zohledňují informace o jaké je možno přijít. Confidentiality Impact je o tom, zda útočník je schopen, díky zranitelnosti, získat přístup k důvěrným datům. Větší riziko bude, pokud se takto útočník dostane až k informacím o všech zaměstnancích a nižší, jestliže se tak dostane pouze k jedné skupině zaměstnanců nebo k žádné. Podkategorie Integrity Impact rozlišuje, zda je útočník schopen narušit autentičnost dat. Pokud je schopen měnit data, bude ohodnocení závažnosti vyšší. U Availability Impact je počítáno skóre na základě toho, zda je útočník schopen zcela odepřít přístup k napadeným datům.

Rozsah, v tomto případě lépe Scope, je podskupina, která je novinkou ve verzi 3.0. Tato metrika počítá, zda je útočník schopen napáchat škody mimo rozhraní vytvořené uživatelem. Např. pokud uživatel spustí aplikaci v sandboxu a útočník bude schopen, díky zranitelnosti, překonat sandbox a napáchat škody v OS, na kterém sandbox běží, je nebezpečnost zranitelnosti daleko vyšší, než pokud škoda bude napáchána pouze v sandboxu.

Další velkou skupinou je Temporal Metric Group neboli časové metriky. Sem se řadí podkategorie Exploit Code Maturity, Remediation Level a Report Confidence. Exploit Code Maturity zvyšuje či snižuje riziko na základě pravděpodobnosti. Jestliže může zranitelnost zneužít kdokoliv a existuje k ní kód, který jí dokáže automaticky využít, je riziko vysoké. Na druhou stranu, pokud je zranitelnost využitelná pouze hypoteticky a útok závisí na prostředí, je riziko malé. Remediation Level je pak podkategorie, kde se nebezpečnost zvyšuje či snižuje na základě četnosti oprav. K nově vydanému softwaru se většinou

pravidelně vydávají opravy, které opravují nebezpečná místa. Oproti tomu starší software již nemusí opravy dostávat a tím se zvyšuje nebezpečnost jeho používání. Poslední podkategorií je Report Confidence. Tato podkategorie vypočítává hodnotu nebezpečnosti zranitelnosti tak, že je uživatel schopen říct s jistotou, že zranitelnost existuje a je v jeho systému. Zde například nejvyšší ohodnocení dostane zranitelnost, která je zdokumentována a je potvrzena. Nižší si pak odnese zranitelnost, kterou někdo nahlásil a neposkytl dostatečné informace.

Poslední velkou skupinou jsou Environmental Metrics, metriky prostředí. Zde jsou uvedeny 3 podobné podkategorie jako v základní skupině. Jsou to Confidentiality, Integrity a Availability Requirement. Zde jsou ovšem tyto podkategorie brány ze subjektivního hlediska. Pro správce webového fóra o psech bude mít zranitelnost, při které přijde o databázi uživatelů, pravděpodobně menší dopad než na správce webového serveru nadnárodní firmy s informacemi o zakázkách. Tuto subjektivní stránku věci je možné zohlednit právě v těchto podkategoriích. Poslední podkategorií je Modified Base Metrics. Ta je podobná jako 3 předešlé. Zde je možné nastavit, v jakém prostředí uživatel skutečně pracuje a může tak zvyšovat nebo snižovat nebezpečnost. Např. pokud zranitelnost potřebuje k odposlouchávání portů administrátorská práva, ale správce si defaultně nastavil práva nižší, může tak upravit výsledné skóre nebezpečnosti zranitelnosti.

Nastavením několika těchto parametrů uživatel dostane číslo, jakým je zranitelnost ohodnocena na základě nebezpečnosti. Od 10 – 9 jako kritická, 8,9 - 7 vysoká, 6,9 - 4 střední, 3,9 - 0,1 nízká a 0 žádná nebezpečnost. Tyto výsledné hodnoty může každý skenovací nástroj interpretovat trochu odlišně. Např. OpenVAS používá pouze vysoká, střední, nízká a žádná.

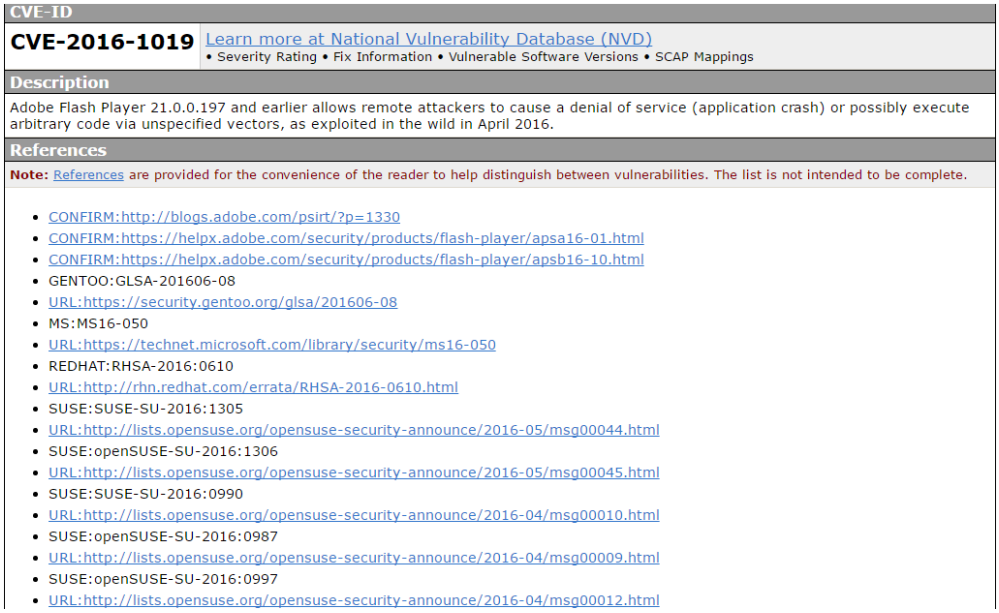
Dané skóre například používá databáze NVD nebo CVE Details. Na oficiální stránce CVSS je i kalkulačka, díky které je možné určit skóre zranitelnosti.

5 Databáze zranitelností

Tato kapitola se zabývá především, představením nejpoužívanějších databází pro nástroje spojené s VM.

5.1 Common Vulnerabilities and Exposures

CVE je, jak sami autoři tvrdí, slovník známých zranitelností. Dříve každá bezpečnostní aplikace používala svou databázi. To mohlo vést k nejasnostem, kdy jedna a tatáž zranitelnost, mohla být v každé databázi zařazena jinak. Právě díky rozšířenosti a kumulaci informací o zranitelnosti, je CVE cenný pomocník při řešení [10]. Záznamy obsahují přidělené CVE-ID, popis a odkazy na stránky, kde je možné najít opravu nebo se o zranitelnosti dočíst více. Databáze je opravdu rozsáhlá a v současné době obsahuje přes 80 000 záznamů. Pro ukázku byla vybrána zranitelnost CVE-2016-1019, která v roce 2016 napadala počítače skrz Adobe Flash Player [11].



CVE-ID

CVE-2016-1019 Learn more at National Vulnerability Database (NVD)
• Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

Description

Adobe Flash Player 21.0.0.197 and earlier allows remote attackers to cause a denial of service (application crash) or possibly execute arbitrary code via unspecified vectors, as exploited in the wild in April 2016.

References

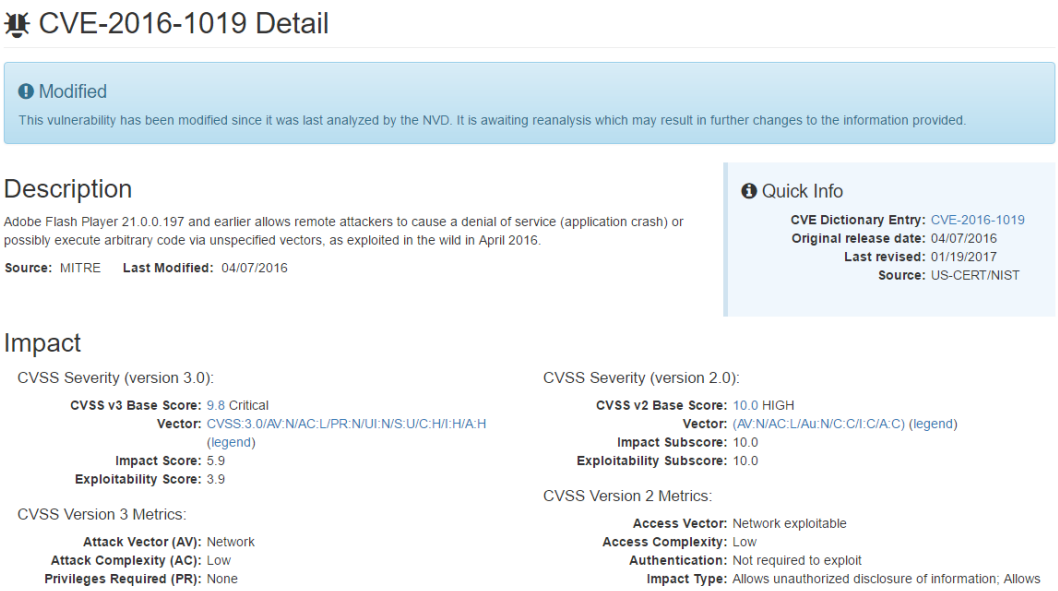
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- CONFIRM:http://blogs.adobe.com/psirt/?p=1330
- CONFIRM:https://helpx.adobe.com/security/products/flash-player/apsa16-01.html
- CONFIRM:https://helpx.adobe.com/security/products/flash-player/apsb16-10.html
- GENTOO:GLSA-201606-08
- URL:https://security.gentoo.org/glsa/201606-08
- MS:MS16-050
- URL:https://technet.microsoft.com/library/security/ms16-050
- REDHAT:RHSA-2016:0610
- URL:http://rhn.redhat.com/errata/RHSA-2016-0610.html
- SUSE:SUSE-SU-2016:1305
- URL:http://lists.opensuse.org/opensuse-security-announce/2016-05/msg00044.html
- SUSE:openSUSE-SU-2016:1306
- URL:http://lists.opensuse.org/opensuse-security-announce/2016-05/msg00045.html
- SUSE:SUSE-SU-2016:0990
- URL:http://lists.opensuse.org/opensuse-security-announce/2016-04/msg00010.html
- SUSE:openSUSE-SU-2016:0987
- URL:http://lists.opensuse.org/opensuse-security-announce/2016-04/msg00009.html
- SUSE:openSUSE-SU-2016:0997
- URL:http://lists.opensuse.org/opensuse-security-announce/2016-04/msg00012.html

Obrázek 4 Ukázka z CVE databáze – printscreen [10]

5.2 National Vulnerability Database

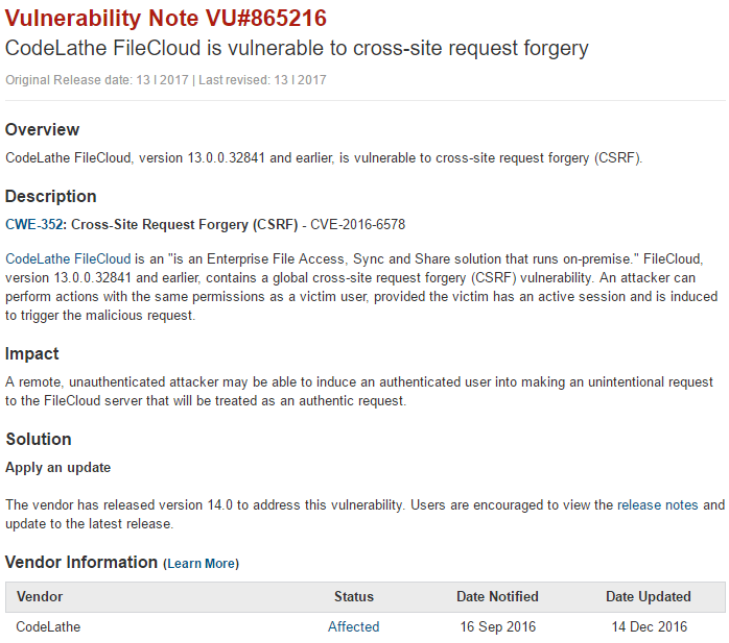
NVD úzce spolupracuje s CVE. Na samotných stránkách CVE je odkaz pro hledání v NVD. Tato databáze je zdroj americké vlády pro standardy ohledně zranitelností [12]. V samotných záznamech je navíc, oproti CVE, seznam verzí produktů, kterých se zranitelnost týká a hodnocení zranitelnosti, který využívá systém hodnocení CVSS 3.0 a 2.0.



Obrázek 5 Ukázka z databáze NVD – printscreen z [12]

5.3 CERT

Společnost CERT se snaží klást důraz na bezpečnost již během vývoje aplikace a věří, že tím je možné detekovat, eliminovat a později se již zcela vyhnout zranitelnostem, ještě předtím, než je aplikace vydána. Toho chtějí dosáhnout poskytováním znalostí, technik a nástrojů pro vývojáře, aby jim pomohli porozumět, jak jsou zranitelnosti vytvářeny a objevovány a zároveň jak se jim vyvarovat [13]. Poskytované nástroje například testují různé vstupy do aplikace a sledují její chování. Společnost CERT také poskytuje databázi zranitelností s řešením, CVSS metrikou a CVE označením.



Vulnerability Note VU#865216

CodeLathe FileCloud is vulnerable to cross-site request forgery

Original Release date: 13 I 2017 | Last revised: 13 I 2017

Overview

CodeLathe FileCloud, version 13.0.0.32841 and earlier, is vulnerable to cross-site request forgery (CSRF).

Description

CWE-352: Cross-Site Request Forgery (CSRF) - CVE-2016-6578

CodeLathe FileCloud is an "is an Enterprise File Access, Sync and Share solution that runs on-premise." FileCloud, version 13.0.0.32841 and earlier, contains a global cross-site request forgery (CSRF) vulnerability. An attacker can perform actions with the same permissions as a victim user, provided the victim has an active session and is induced to trigger the malicious request.

Impact

A remote, unauthenticated attacker may be able to induce an authenticated user into making an unintentional request to the FileCloud server that will be treated as an authentic request.

Solution

Apply an update

The vendor has released version 14.0 to address this vulnerability. Users are encouraged to view the release notes and update to the latest release.

Vendor Information (Learn More)

Vendor	Status	Date Notified	Date Updated
CodeLathe	Affected	16 Sep 2016	14 Dec 2016

Obrázek 6 Ukázka z CERT databáze – printscreen z [13]

5.4 Exploit DB

Tato databáze se oproti ostatním liší tím, že neposkytuje návod, jak chybu opravit, ale naopak jak jí zneužít. Na to se dá pohlížet tak, že dává možnost napadení aplikací nebo webu, ale na stranu druhou se jedná o zdokumentovaný postup, jak zranitelnost zneužít a odborníci můžou pracovat na opravě. Záznamy mohou obsahovat i CVE číslo. Exploity jsou testovány a často je k dispozici i zdrojový kód [14].

Adobe Flash Player 24.0.0.186 - 'ActionGetURL2' Out-of-Bounds Memory Corruption (1)

EDB-ID: 41008	**Author**: COSIG	**Published**: 2017-01-11
CVE: CVE-2017-2930	**Type**: Dos	**Platform**: Multiple
Aliases: N/A	**Advisory/Source**: Link	**Tags**: Vulnerability, Denial of Service (DoS)
E-DB Verified: ✔	**Exploit**: Download / View Raw	**Vulnerable App**: N/A

« Previous Exploit Next Exploit »

```
Source: https://cosig.gouv.qc.ca/en/cosig-2017-01-en/

################################################################################

# Application: Adobe Flash Player
# Platforms: Windows,OSX
# Versions: 24.0.0.186 and earlier
# Author: Francis Provencher of COSIG
# Website: https://cosig.gouv.qc.ca/en/advisory/
# Twitter: @COSIG_
# Date: January 10, 2017
# CVE-2017-2930
# COSIG-2016-35

################################################################################

1) Introduction
2) Report Timeline
3) Technical details
4) POC

################################################################################

===============
1) Introduction
```

Obrázek 7 Ukázka z Exploit DB – printscreen z [14]

6 Kategorie nástrojů

Pro rozdělení nástrojů byla, jako hlavní kritérium vybrána licence, pod kterou jsou programy šířeny. Do sestavení systému jsou vybrány ty programy, které jsou šířeny pod licencí GNU/GPL a jí podobné, např. MIT, LGPL atd. Pro ukázku a porovnání jsou do srovnávací části zařazeny i komerční nástroje.

6.1 Svobodný software

Za první zmínku stojí OpenVAS. Tento skenovací program je spravován skupinou dobrovolníků. Mnozí z nich jsou IT odborníky na bezpečnost. Nástroj je šířen pod licencí GNU/GPL. Nabízí množství nastavení, díky kterým si uživatel může nastavit skenovaní podle vlastních potřeb. Nebyl nalezen jiný stejně dobrý a podobně vybavený nástroj s licencí GNU/GPL.

Nmap je program, který skenuje porty zvolených IP adres. Díky množství přepínačů je možné skenovat porty UDP i TCP, je možné zjistit OS, který běží na dané IP adrese nebo verzi služeb, které na daném portu běží. Nmap je opravdu schopný nástroj.

Dalším podobným programem jako je Nmap je Netcat. Netcat dokáže skenovat porty, sledovat provoz na portu nebo port forwarding, kdy Netcat na jednom portu naslouchá a když je port aktivní přesměruje komunikaci na jiný.

Do svobodných nástrojů je zahrnut i program W3af. Cílem tohoto projektu je vytvořit framework pro testování bezpečnosti webové aplikace. Nástroj nabízí ovládání přes příkazovou řádku, ale i grafické rozhraní. V samotném nástroji je možné nastavit jaké pluginy budou použity pro otestování stránky, např. plugin sqli se stará o testování techniky SQL injection. Kompletní nastavení je pak možné uložit do profilů. W3af je šířen pod licencí GNU/GPL a je napsán v jazyce Python 2.7.

Podobným nástrojem jako W3af je Nikto2. Nikto je napsán v jazyce Perl. Bohužel už je dnes jeho vývoj zastaven, přesto má co nabídnout. Dokáže najít špatnou konfiguraci web serveru, staré verze programů nebo nezabezpečené soubory na serveru. Nikto nenabízí takové možnosti nastavení jako W3af. Šířen je pod licencí GNU/GPL.

6.2 Komerční software

Do této skupiny byly zařazeny dva nástroje, které nabízí i trial verze. Prvním takovým je Nexpose od Rapid7. K dispozici jsou dvě verze – Community a Enterprise. Community Edition je omezená na skenování 32 IP adres, Enterprise Edition je potom neomezená. Tento nástroj dokáže skenovat nejenom fyzická zařízení, ale i virtuální, cloudová a mobilní. Nástroj hodnotí zranitelnosti podle vlastního hodnotícího systému, jenž má rozsah od 1 do 1000, ale je možné se podívat i na klasické CVSS hodnocení. Firma Rapid7 vyvíjí i nástroj Metasploit, který se zabývá penetračním testováním. S použitím těchto dvou nástrojů dohromady, lze dosáhnout silného zabezpečení sítě [15].

Druhým nástrojem je Nessus od firmy Tenable. Jedná se o skenovací nástroj. Ten je nabízen ve dvou verzích, a to Professional a Manager. Verze Professional je určená pro jedince a verze Manager pro společnosti. Obě nabízí 7-denní trial verzi na vyzkoušení. Nástroj nabízí skenování zranitelností, detekci malwaru, skenování webových aplikací a další. Nessus se také chlubí podporou široké škály síťových zařízení, operačních systémů, databází a aplikací, ať už na fyzickém, virtuálním nebo cloudovém zařízení [16].

6.3 Software as a Service

Software as a Service (SaaS) je typ softwaru, kdy uživatel by neměl nic instalovat a komunikace probíhá přes webovém rozhraní. Právě kvůli absenci fyzického zařízení se firma nemusí starat o nákup hardwaru, instalaci softwaru a případnou údržbu. Sestavení a nakonfigurování takového stroje může být drahá záležitost, a tak jsou některé společnosti nakloněny právě tomuto modelu služby. SaaS je také oproštěn od aplikace aktualizací, protože vždy bude k dispozici nejnovější verze. Nedostatky jsou pak pravidelné poplatky a také spoléhání se na třetí stranu, kdy tato strana má výsledky ze skenů.

Qualys FreeScan je nástroj využívající právě tento model. Firma Qualys je významnou společností zabývající se bezpečností IT. FreeScan nabízí nejenom oskenování a nalezení zranitelností na síti, ale kontroluje i webové aplikace. Nástroj nabízí i neustálý monitoring, kdy nástroj upozorní na nová zařízení při připojení do sítě nebo při detekci zranitelnosti s vysokým ohodnocením nebezpečnosti. Mezi důležité funkce se řadí i vizualizace sítě [17].

7 Praktická část

7.1 Specifikace systému pro Vulnerability Management

V první řadě je nutné specifikovat požadavky tvořeného systému pro VM v této práci. Vybrané požadavky jsou:

- pravidelné denní skenování portů zadaných IP adres,
- ukládání logů z denního skenování do textového souboru a databáze,
- porovnávání logů, zda byl nalezen nový port,
- pravidelné měsíční skenování zranitelností,
- ukládání logů z měsíčního skenování do databáze,
- skenování zadané webové aplikace na požadavek,
- ukládání logů ze skenování webové aplikace do databáze.

7.2 Testovací systém

Testování bylo prováděno na virtuálním systému, pomocí VirtualBoxu od firmy Oracle. Hardwarové specifikace vytvořeného stroje jsou:

- procesor – Intel Core i7-3612QM, 2.10 GHz, používaná 4 jádra,
- pamět RAM – 4 GB,
- grafická karta – vytvořená programem s pamětí o velikosti 128 MB,
- ostatní hardware byl vytvořen VirtualBoxem.

Pro základ praktické části je vybrán OS Ubuntu. Zvolen je na základě podpory ze strany používaných nástrojů. Ubuntu pracuje na linuxovém jádře a v době psaní práce je dostupná verze 16.04 – Xenial Xerus. Ubuntu se hlásí k open source systémům s licencí, která dává k dispozici zdrojový kód, je možné ho měnit, šířit a je volně ke stažení.

Při instalaci je uživatel dotázán pouze na to nejnutnější. Jako jazyk byla zvolena angličtina. Dalším krokem je povolení instalace aktualizací již během instalace. Aplikace třetích stran, jako MP3 přehrávač, jsou zbytečné, a tak jsou vynechány. Instalátor sám zformátuje disk na souborové systémy ext3 a swap, kdy ext3 je souborový systém používaný v distribucích linuxu a swap je část disku pro rozšíření virtuální paměti. Následuje zvolení časového pásma a vytvoření uživatele. Po vytvoření byly staženy nejnovější balíčky. Všechny operace probíhají pod účtem správce.

```
apt-get update
apt-get upgrade
reboot
```

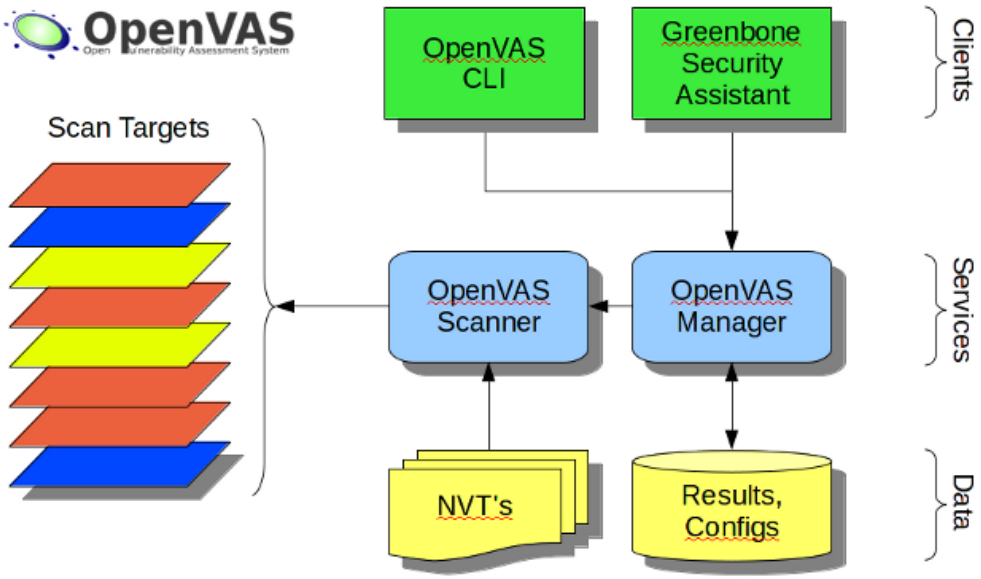
Testovací skenování bude probíhat jak na vytvořeném virtuálním PC, tak na vytvořeném serveru pomocí virtuálního systému Metasploitable 2. Metasploitable je linuxová distribuce na bázi Ubuntu, záměrně vytvořená tak, aby obsahovala chyby a poskytovala testovací zařízení pro nástroje hledající zranitelnosti. Nejdříve je nutné upravit síťová rozhraní ve VirtualBoxu. Místo defaultního NAT nastavení, které oběma virtuálním strojům přiřadí stejnou IP adresu, je zvolena možnost NAT network. Ta každému stroji přiřadí unikátní vnitřní IP adresu a zároveň na sebe oba vidí. Po nabootování a zadání jména a hesla se spustí webový server na systému Metasploitable. Nyní je pro testované nástroje, vytvořeno ideální testovací rozhraní. Pro první sken je volena IP adresa vytvořeného virtuálního PC a pro další skeny, pak IP adresa systému Metasploitable. IP adresy jsou přidělovány dynamicky, takže je bezpředmětné je uvádět.

7.3 Testy nástrojů

V této kapitole je otestováno několik nástrojů. Cílem je vybrat open source programy, které budou použity při tvorbě VM. Komerční programy jsou zahrnuty z důvodu porovnání účinnosti s open source programy. První sken je spuštěn na vytvořeném virtuálním PC, kde je v provozu Apache server na portu 80, dále GSA manažer na portu 443, na portu 25 naslouchá SMTP protokol a portu 9390 využívá OpenVAS pro ovládání manažera. Skeny jsou nastavené na skenování bez webového prohledávání, pokud to jde. Druhý sken probíhá na systému Metasploitable. Tyto skeny jsou nastaveny pro skenování i webového rozhraní.

7.3.1 OpenVAS

OpenVAS je volně dostupný nástroj. Podle oficiálních stránek [18], jde o framework, který spojuje několik nástrojů a služeb a nabízí tak obsáhlý a silný nástroj pro základ VM. Skener využívá protokol SCAP. SCAP je protokol, který standardizuje přístup k provádění údržby bezpečnosti systému, a to skrze sadu bezpečnostních pravidel. Protokol byl přijat jako norma NIST SP 800-117. Nejpodstatnější částí je Network Vulnerability Tests, jelikož právě díky těmto testům je nástroj schopen rozpoznávat jednotlivé zranitelnosti.



Obrázek 8 Schéma OpenVAS - převzato z [18]

Tento nástroj se skládá z několika důležitých prvků. Jádrem architektury je skener, který se synchronizuje s databází testů. Synchronizace probíhá skrze OpenVAS NVT Feed, což je proces zajišťující online aktualizaci testů. Dalším podstatným prvkem je OpenVAS Manager. Manager ovládá skener skrz OpenVAS Transfer Protocol (OTP). Na manager je možné naimplementovat různorodé klienty pro filtrování nebo řazení výsledků skenů, v této práci je použit Greenbone Security Assistant (GSA), což je webové rozhraní, pracující na portu 443 - HTTPs. Manager ovládá i SQL lite databázi, kde se ukládají veškerá data a nastavení. Manager nabízí i user management pro přidávání uživatelů a přidělování práv. Poslední podstatnou součástí je OpenVAS CLI, příkazový řádek, který ovládá OMP. OMP je OpenVAS Management Protocol, což je XML protokol pro ovládání managera.

Na instalaci do Ubuntu jsou nutné balíčky, které nejsou v základu systému. Balíček sqlite3 zajišťuje podporu relačních databází, druhý balíček potom umožňuje instalovat z repozitáře ppa.

```
apt-get install sqlite3 software-properties-common
```

Pro stáhnutí a instalaci OpenVAS je nutné přidat do zdrojů projekty z repozitáře ppa. Následuje aktualizace balíčků.

```
add-apt-repository ppa:mrazavi/openvas
apt-get update
apt update
```

Nyní již je vše připraveno pro samotnou instalaci. Tu provedeme následujícím příkazem.

```
apt-get install openvas
```

Dále je provedena synchronizace s databázemi, potřebných pro správnou práci programu OpenVAS. Tyto procesy trvají poměrně dlouho.

```
openvas-nvt-sync
openvas-scapdata-sync
openvas-certdata-sync
```

Následuje restart služeb.

```
service openvas-scanner restart
service openvas-manager restart
```

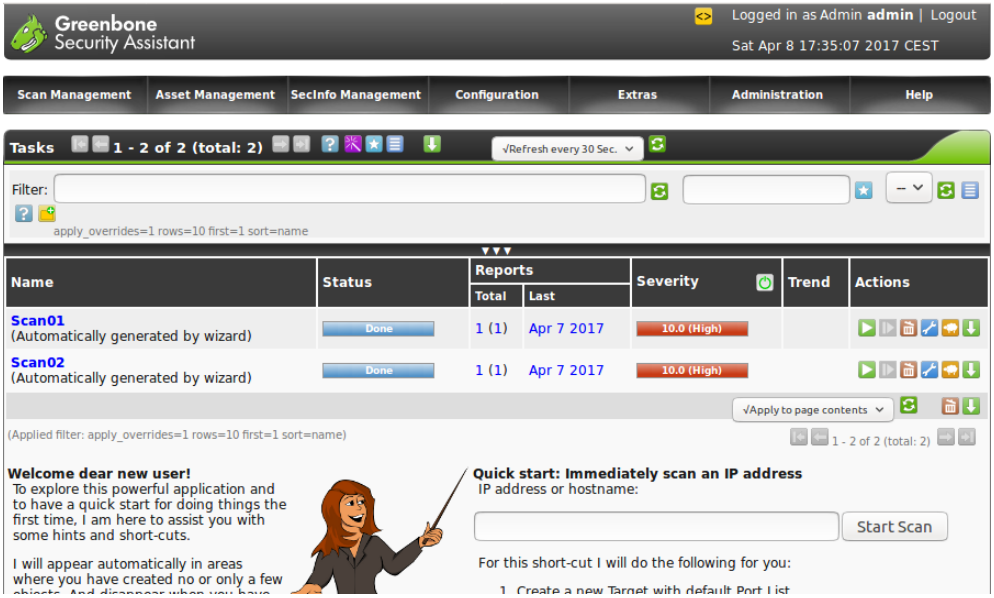
Příkaz níže aktualizuje vytvořenou databázi.

```
openvasmd --rebuild --progress
```

Jako poslední je povolen port 443, protože webový manažer běží, právě na tomto portu.

```
ufw allow https
```

Nyní je celý proces u konce a OpenVAS je přípraven k práci. Pro správně fungování je nutné, alespoň jednou týdně aktualizovat používané databáze. Pro připojení je možné zadat do adresy v prohlížeči https://IP_adresa:443 popř. https://localhost:443. Zde je možnost provádět testy, aktualizovat databáze, přiřazovat přístupová práva a další.



Obrázek 9 OpenVAS - printscreen

Program poskytuje různé sady testů. Full and fast nabízí poměrně rychlý sken, kdy při své práci může využívat již provedených testů. Verze ultimate provede test, kdy je možné, že budou některé služby resetovány nebo vypnuty. Full and very deep je typ testu, který zabere více času, ale nepracuje s informacemi z logů proběhlých v minulosti. Verze ultimate je pak stejná jako u full and fast. Pro sken je vybrán full and very deep.

První sken našel pouze 4 zranitelnosti. Jednu vážnou, a to sice přihlašovací údaje do GSA manažera, a další 3 středně závažné.

Vulnerability	Severity	QoD	Host	Location	Actions
GSA Default Admin Credentials	10.0 (High)	100%	10.0.2.4	443/tcp	
Apache /server-status accessible	5.0 (Medium)	99%	10.0.2.4	80/tcp	
SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	5.0 (Medium)	98%	10.0.2.4	443/tcp	
SSL/TLS: Diffie-Hellman Key Exchange Insufficient DH Group Strength Vulnerability	4.0 (Medium)	80%	10.0.2.4	3780/tcp	
OS Detection Consolidation and Reporting	0.0 (Log)	80%	10.0.2.4	general/tcp	
Traceroute	0.0 (Log)	80%	10.0.2.4	general/tcp	
CPE Inventory	0.0 (Log)	80%	10.0.2.4	general/CPE-T	
HTTP Server type and version	0.0 (Log)	80%	10.0.2.4	80/tcp	
phpPgAdmin Detection	0.0 (Log)	80%	10.0.2.4	80/tcp	
Services	0.0 (Log)	80%	10.0.2.4	80/tcp	
CGI Scanning Consolidation	0.0 (Log)	80%	10.0.2.4	80/tcp	
Apache Web Server Version Detection	0.0 (Log)	80%	10.0.2.4	80/tcp	
Services	0.0 (Log)	80%	10.0.2.4	443/tcp	
Services	0.0 (Log)	80%	10.0.2.4	443/tcp	
SSL/TLS: Report Non Weak Cipher Suites	0.0 (Log)	98%	10.0.2.4	443/tcp	
SSL/TLS: Collect and Report Certificate Details	0.0 (Log)	98%	10.0.2.4	443/tcp	
Greenbone Security Assistant Detection	0.0 (Log)	80%	10.0.2.4	443/tcp	

Obrázek 10 OpenVAS – printscreen

Druhý sken našel o poznání více chyb, celkem 69. Konkrétně 22 závažných, 43 středních a 4 méně závažné zranitelnosti.

Vulnerability	Severity	QoD	Host	Location	Actions
OS End Of Life Detection	10.0 (High)	80%	10.0.2.6	general/tcp	
TWiki XSS and Command Execution Vulnerabilities	10.0 (High)	80%	10.0.2.6	80/tcp	
Check for rexecd Service	10.0 (High)	80%	10.0.2.6	512/tcp	
Possible Backdoor: Ingreslock	10.0 (High)	99%	10.0.2.6	1524/tcp	
X Server	10.0 (High)	80%	10.0.2.6	6000/tcp	
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99%	10.0.2.6	8787/tcp	
DistCC Remote Code Execution Vulnerability	9.3 (High)	99%	10.0.2.6	3632/tcp	
SSH Brute Force Logins With Default Credentials Reporting	9.0 (High)	95%	10.0.2.6	22/tcp	
MySQL / MariaDB weak password	9.0 (High)	95%	10.0.2.6	3306/tcp	
PostgreSQL weak password	9.0 (High)	99%	10.0.2.6	5432/tcp	
DistCC Detection	8.5 (High)	95%	10.0.2.6	3632/tcp	
PostgreSQL Multiple Security Vulnerabilities	8.5 (High)	80%	10.0.2.6	5432/tcp	
vsftpd Compromised Source Packages Backdoor Vulnerability	7.5 (High)	99%	10.0.2.6	21/tcp	
phpMyAdmin Code Injection and XSS Vulnerability	7.5 (High)	80%	10.0.2.6	80/tcp	
phpMyAdmin BLOB Streaming Multiple Input Validation Vulnerabilities	7.5 (High)	80%	10.0.2.6	80/tcp	
phpMyAdmin Configuration File PHP Code Injection Vulnerability	7.5 (High)	80%	10.0.2.6	80/tcp	

Obrázek 11 OpenVAS – printscreen

Skeny trvaly průměrně 8-10 minut. Testy typu full and very deep skenují většinu zranitelností, které poskytují NVT. Tedy jak ty bezpečnostní na PC, tak ty webové.

7.3.2 W3af

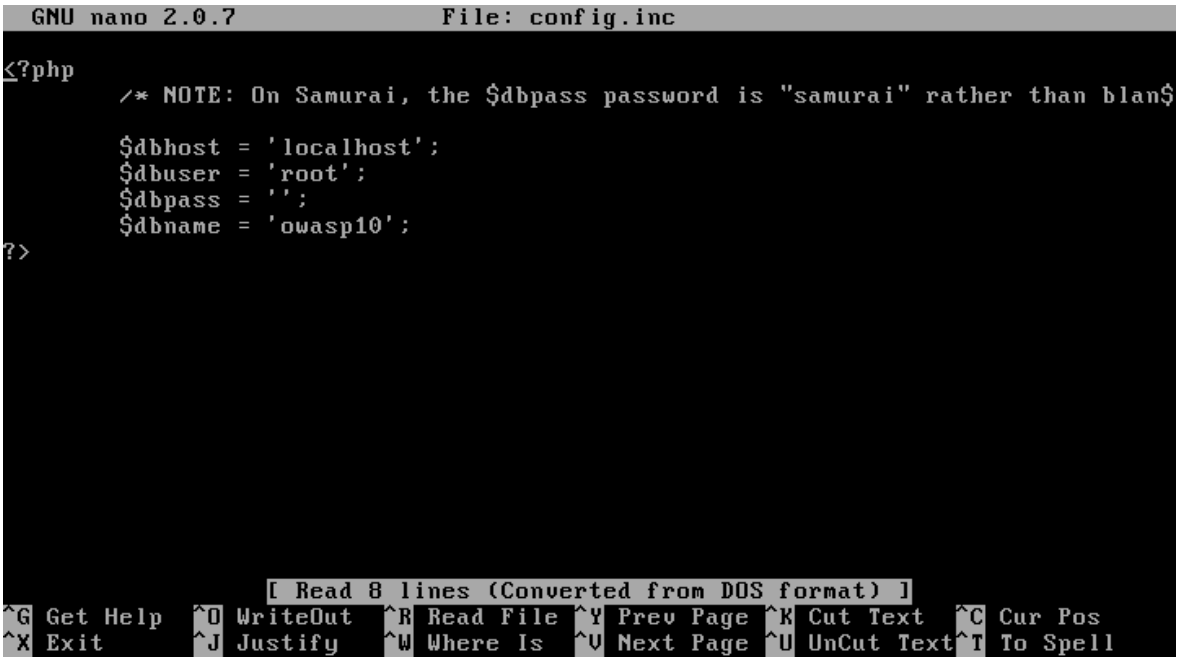
Jak již bylo uvedeno výše, program potřebuje ke svému spuštění Python 2.7. Ten je na Ubuntu nainstalován již v základu. Dále je nutné nainstalovat git, což je distribuovaný systém pro správu verzí programu a také python-pip, jehož hlavní práce je instalovat moduly jazyku Python. Poté je naklován repozitář s knihovnou w3af. Nakonec je spuštěna aplikace z konzole. Je možné, že pro správnou práci je potřeba doinstalovat závislosti, a proto je v adresáři tmp, vytvořen skript pro tyto účely.

```
apt install git
apt-get install python-pip
pip install --upgrade pip
git clone --depth 1 https://github.com/andresriancho/w3af.git
cd ~/w3af
```

```
./w3af_console
cd /tmp/
./w3af_dependency_install.sh
```

Ještě před spuštěním testování je potřeba opravit chybu v systému Metasploitable. Stránka s přihlašováním do webové aplikace mutillidae je postižena chybou. Mutillidae je webová aplikace určená k testování. Stačí v systému upravit soubor config.inc a přepsat hodnotu dbname z metasploit na owasp10.

```
cd /var/www/mutillidae
sudo nano config.inc
```



Obrázek 12 Metasploitable úprava - printscreen

Nyní je již možné bez problémů spustit skripty w3af. Ve složce w3af jsou 3. W3af_api, w3af_console a w3af_gui. Pro ukázku i v samotném systému, je pracováno s w3af_console. Console pracuje pouze s příkazovou řádkou. Po spuštění je třeba nastavit několik věcí pro správnou práci. První z nich je profiles. Toto nastavení poskytuje několik základních profilů pro skenování webové aplikace. Zvolen je profil empty_profile, protože dovoluje uživateli určit si vlastní nastavení. Další a nejdůležitější částí nastavení je plugins. V této části je uživateli dovoleno si libovolně konfigurovat pluginy, které použije na webovou stránku. Pro otestování je vybrána přihlašovací stránka webové aplikace mutillidae na sql injection a xss. Testy na sql injection a xss jsou vybrány z důvodu nabídky jak ve W3af, tak v Nikto. Proto

jsou jako pluginy zvoleny blind_sqli, sqli a xss z nabídky. Nastavení konfigurace only_forward u web_spider zajišťuje, že prohledány jsou pouze odkazy v zadané stránce. Jako výstup je požadován report v html formátu. Toho je docíleno povolením html_file v nastavení plugins - output.

```
./w3af_console
w3af >>> profiles
w3af/profiles >>> use empty_profile
w3af >>> back
w3af >>> plugins
w3af/plugins >>> audit blind_sqli sqli
                 htaccess_methods xss
w3af/plugins >>> crawl web_spider
w3af/plugins >>> crawl config web_spider
w3af/plugins >>> set only_forward TRUE
w3af >>> back
w3af/plugins >>> output html_file
```

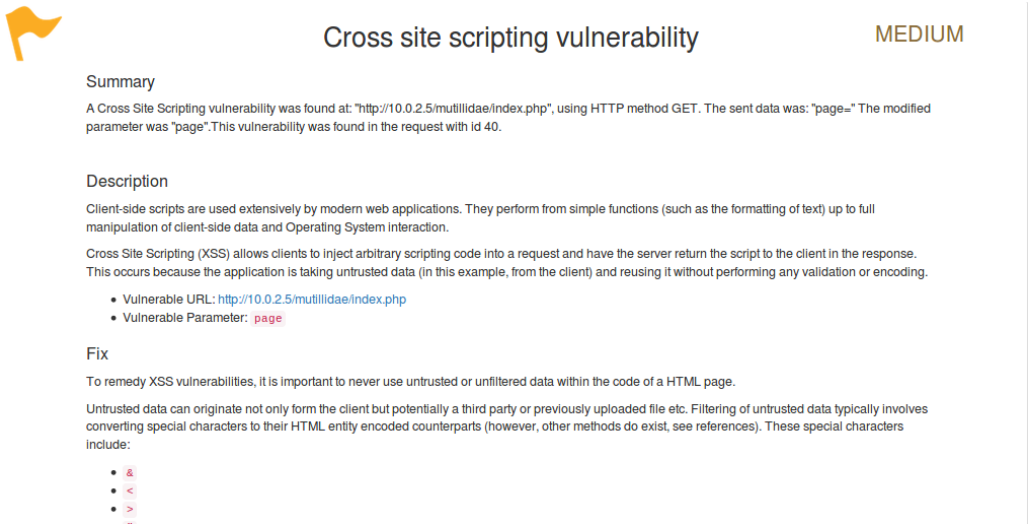
Nyní je možné specifikovat cíl.

```
w3af >>> target
w3af/config:target >>> set target
http://10.0.2.5/mutillidae/index.php?page=login.php
w3af/config:target >>> back
```

V tuto chvíli je vše připraveno pro sken. Stačí pouze spustit.

```
w3af >>> start
```

Vytvořený report je možné zobrazit ve webovém prohlížeči firefox. Nalezeny jsou 4 zranitelností pomocí sql injection a 10 zranitelností pomocí techniky xss. Testy probíhaly kolem 10-15 minut.



Cross site scripting vulnerability MEDIUM

Summary

A Cross Site Scripting vulnerability was found at: "http://10.0.2.5/mutillidae/index.php", using HTTP method GET. The sent data was: "page=" The modified parameter was "page".This vulnerability was found in the request with id 40.

Description

Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction.

Cross Site Scripting (XSS) allows clients to inject arbitrary scripting code into a request and have the server return the script to the client in the response. This occurs because the application is taking untrusted data (in this example, from the client) and reusing it without performing any validation or encoding.

- Vulnerable URL: http://10.0.2.5/mutillidae/index.php
- Vulnerable Parameter: page

Fix

To remedy XSS vulnerabilities, it is important to never use untrusted or unfiltered data within the code of a HTML page.

Untrusted data can originate not only form the client but potentially a third party or previously uploaded file etc. Filtering of untrusted data typically involves converting special characters to their HTML entity encoded counterparts (however, other methods do exist, see references). These special characters include:

- &
- <
- >

Obrázek 13 W3af report - printscreen

7.3.3 Nikto

Při instalaci je nutné stáhnout program z oficiálních stránek a rozbalit.

```
wget https://cirt.net/nikto/nikto-2.1.5.tar.gz
tar xvzf nikto-2.1.5.tar.gz
```

Samotný program je v jazyce Perl. Ten je součástí Ubuntu, a tak není problém se spuštěním.

```
cd nikto-2.1.5
perl nikto.pl
```

Při zadání těchto příkazů se ukáže jednoduchý manuál a je možné začít program používat. Test je stejný jako pro W3af, tedy otestovat aplikaci mutillidae na xss a sql injection. Nutné je specifikovat formát a jméno výstupního souboru. Potřeba je zadat i co se má testovat a jaká webová stránka.

```
perl nikto.pl -T 49 -Format htm -o test.html -host
http://adresa_webove_stranky
```

Výsledný report je nepřehledný, co se týče grafického zpracování. Odkazy na záznamy většinou nefungují. Také není poznat, kdy se jedná o xss, kdy o sql injection. Skeny trvaly pouze pár vteřin. Na místě jsou pochybnosti o správném výsledku.

URI	/mutillidae/index.php/
HTTP Method	TRACK
Description	HTTP TRACK method is active, suggesting the host is vulnerable to XST
Test Links	http://10.0.2.5:80/mutillidae/index.php/ http://10.0.2.5:80/mutillidae/index.php/
OSVDB Entries	OSVDB-877
URI	/mutillidae/index.php/mutillidae/index.php/kboard/
HTTP Method	GET
Description	/mutillidae/index.php/kboard/: KBoard Forum 0.3.0 and prior have a security problem in forum_edit_post.php, forum_post.php and forum_reply.php
Test Links	http://10.0.2.5:80/mutillidae/index.php/mutillidae/index.php/kboard/ http://10.0.2.5:80/mutillidae/index.php/mutillidae/index.php/kboard/
OSVDB Entries	OSVDB-0
URI	/mutillidae/index.php/mutillidae/index.php/lists/admin/
HTTP Method	GET
Description	/mutillidae/index.php/lists/admin/: PHPList pre 2.6.4 contains a number of vulnerabilities including remote administrative access, harvesting user info and more. Default login to admin interface is admin/phplist
Test Links	http://10.0.2.5:80/mutillidae/index.php/mutillidae/index.php/lists/admin/ http://10.0.2.5:80/mutillidae/index.php/mutillidae/index.php/lists/admin/
OSVDB Entries	OSVDB-0

Obrázek 14 Nikto - report

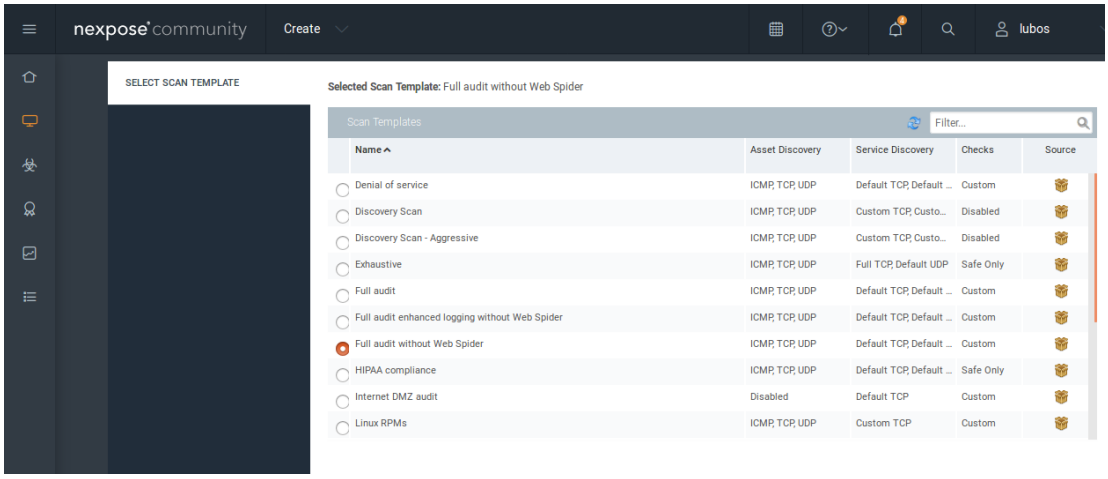
7.3.4 Nexpose Community Edition

Edice Community je značně omezená. Poskytuje pouze skenování pro 32 IP adres. Je zahrnut z důvodu porovnání neplacené trial verze s ostatními nástroji. Cena placené verze pro malé organizace, do 256 IP adres, je stanovena na 3000$ [15].

Po registraci na stránkách společnosti Rapid7 je stažen binární instalační soubor s nástrojem Nexpose Community Edition. Tomuto souboru je potřeba přidat práva na spustitelnost a poté zahájit instalaci.

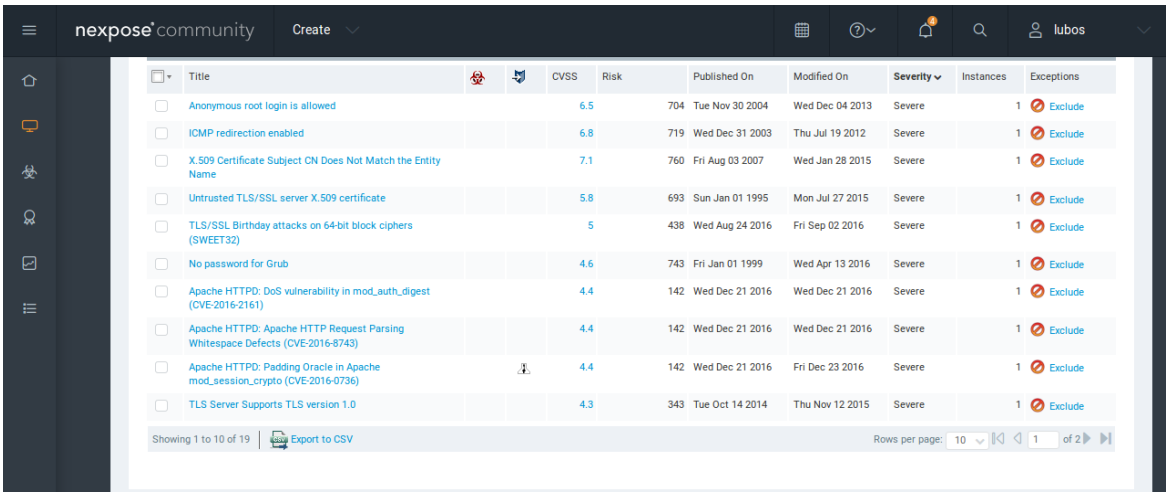
```
chmod +x NeXposeSetup-Linux64.bin
./NeXposeSetup-Linux64.bin
```

V okně, které se objeví, se potvrdí licenční podmínky. V dalším kroku je volba, zda nainstalovat konzoli včetně skenovacího jádra nebo jen jádro. Pro práci je vybrána kompletní instalace. Následuje okno s požadavky, kde pro doporučený běh je požadováno 8 GB RAM a přes 80 GB místa na disku. Je možné zvolit port, na kterém bude probíhat komunikace s databází. Je ponechán port 3780. Další okno nabízí zvolení jména a hesla uživatele a od tohoto momentu se již nástroj instaluje. Po instalaci je možné, na adrese https://localhost:3780, zadat IP adresy, které se budou skenovat a vybrat šablonu skenu. Pro první sken je vybrán Full Audit without Web Spider.



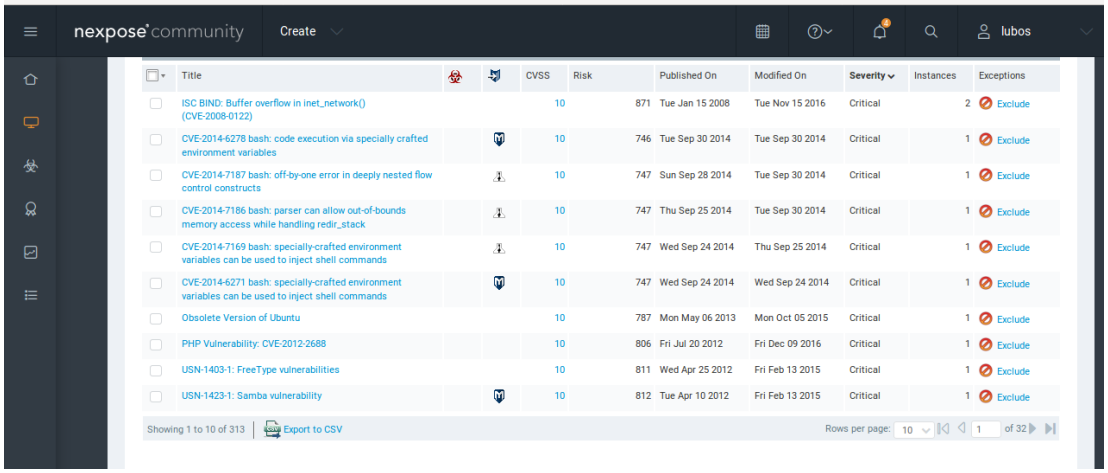
Obrázek 15 Nexpose - printscreen

První sken našel 13 zranitelností, které ohodnotil jako severe a 6 zranitelností s označením moderate. Nexpose používá své vlastní označení pro hodnocení zranitelností založené na CVSS. Skládá se z critical >=8, severe >=4 a moderate <4.



Obrázek 16 Nexpose - printscreen

Druhý sken, který byl nastaven na Full Audit, skenování včetně skenování webového rozhraní, našel zranitelností o poznání více. 313 zranitelností je obrovské číslo, které nenašel žádný jiný skener. Z toho 174, mají dokumentovaný postup ke zneužití.



Obrázek 17 Nexpose - printscreen

Skeny obou adres trvaly průměrně 8 minut.

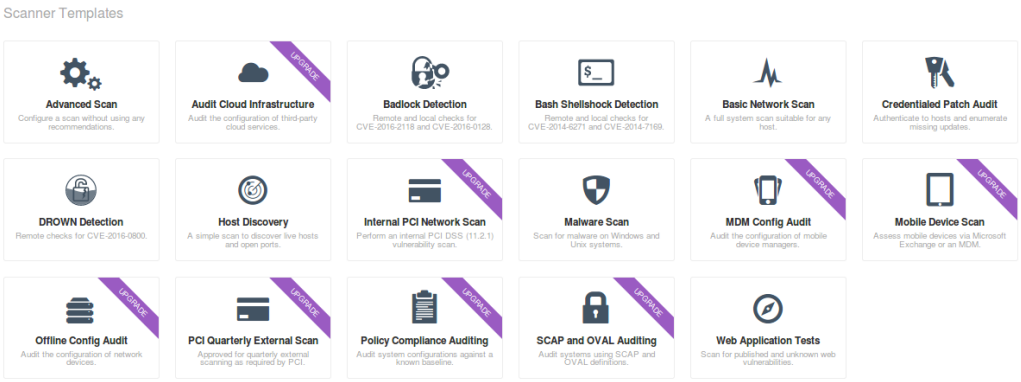
7.3.5 Nessus Professional

Trial verze programu Nessus Professional je omezená na 7 denní používání. Plnou verzi si pak tvůrci cení na 2 190$ ročně. Po stažení je třeba program nainstalovat přes příkaz dpkg, který dokáže pracovat s balíčky typu .deb. Po proklikání se instalací je spuštěn samotný program.

```
dpkg -i Nessus.deb
/etc/init.d/nessusd start
```

Nyní je možné na adrese https://localhost:8834 nalézt webové rozhraní Nessusu. Program se napoprvé poměrně dlouho, v řádech desítkách minut, inicializuje. Při další práci již nebyl proces tak dlouhý. Poprvé se musí zadat aktivační kód, který uživateli přijde emailem.

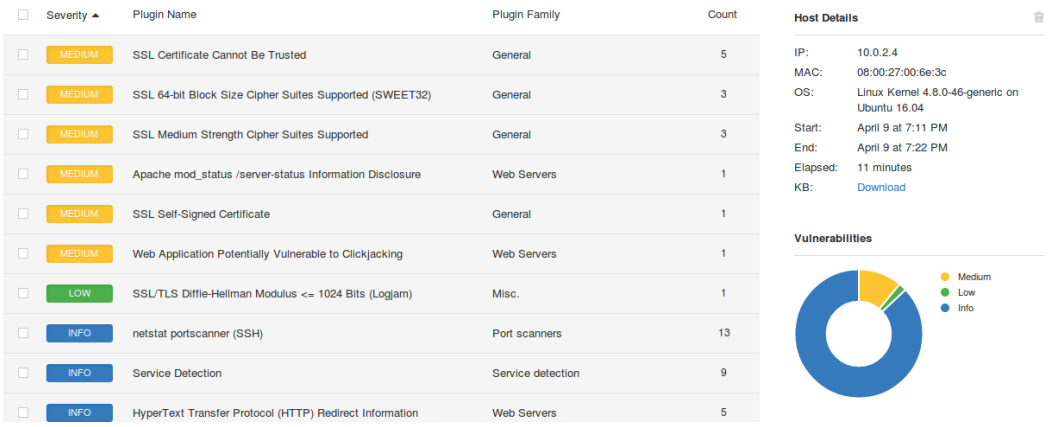
Nástroj nabízí značné množství skenů. Zvolen jest Advanced Scan. Ten nabízí konfiguraci skenování zranitelností jak na počítači, tak i ve webovém rozhraní. V podstatě se jedná o základní sken Basic Network scan s více možnostmi.



Obrázek 18 Nexpose - printscreen

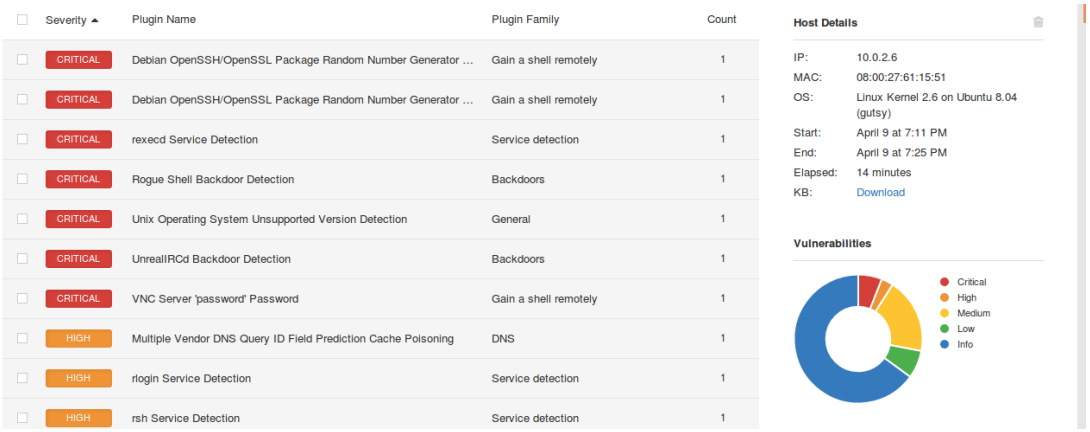
Po zadání skenovaných IP adres je možné nastavit rozvrh v jakých intervalech budou skeny probíhat. Na řadu přichází samotné nastavení skenu. Je možné nastavit, zda skenovat tiskárny, jaké metody ping se mají používat, zda skenovat TCP i UDP porty, jaký rozsah portů skenovat, zda skenovat webové aplikace a další. Po nastavení došlo ke skenování.

První sken našel 6 středních a 1 nízkou zranitelnost. Sken byl nastaven pro skenování pouze virtuálního PC bez webového rozhraní.



Obrázek 19 Nexpose - printscreen

Druhý našel, očekávaně, více zranitelností. Celkem 46. 7 kritických, 4 vysoké, 25 středních a 10 nízkých. Sken proběhl včetně webového rozhraní.



Obrázek 20 Nexpose - printscreen

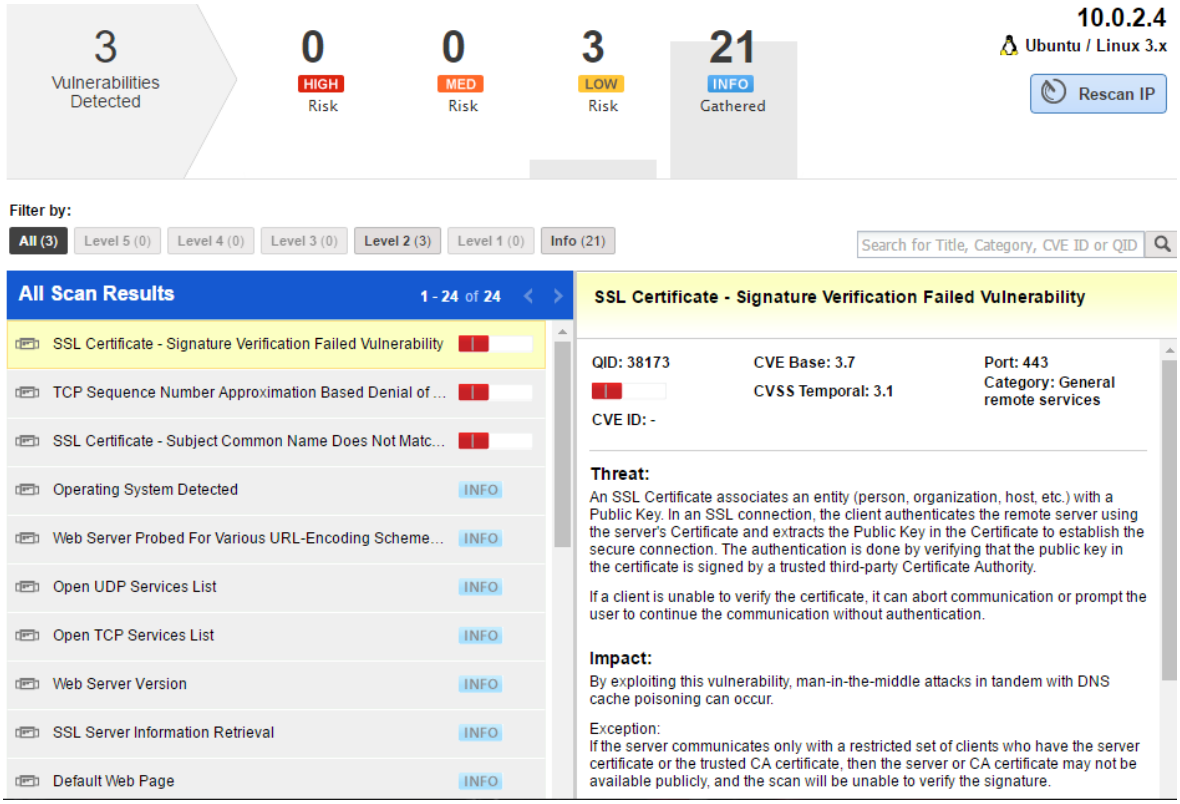
Nessus nabízí, podle názoru autora práce, nejpřehlednější rozhraní. Skenování probíhalo přibližně 14 minut.

7.3.6 Qualys FreeScan Trial

Nástroj Qualys FreeScan Trial je poskytován jako SaaS. Tato verze je opět striktně omezená a to na 10 skenů. Jestliže chce uživatel otestovat PC z vnitřní sítě, je donucen nainstalovat distribuci skeneru do VirtualBoxu. Pro někoho může být problém cena. Pro malé firmy, do 256 IP adres, je její výše stanovena na 795$ za rok [19].

Při pokusu o sken adresy 10.0.2.4 je zobrazena zpráva o skenování IP v rámci lokální sítě a žádost o stažení virtuálního skeneru. Virtuální disk stačí pouze otevřít a naimportuje se. První obrazovka vyžaduje kód, který je vygenerovaný na stránkách Qualys, díky kterému se skener nakonfiguruje. Základní sken je nastaven jak na PC v síti, tak na servery. Není tedy zásadní rozdíl, zda se skenuje PC nebo server. Nastavení je stejné.

První sken proběhl relativně rychle, v řádu minut a našel 3 nízko hodnocené zranitelnosti.



Obrázek 21 Qualys FreeScan - printscreen

Druhý sken běžel přes 18 hodin a stále nebyl hotov. Tato doba by v reálném použití byla naprosto nežádoucí, a proto byl sken zrušen.

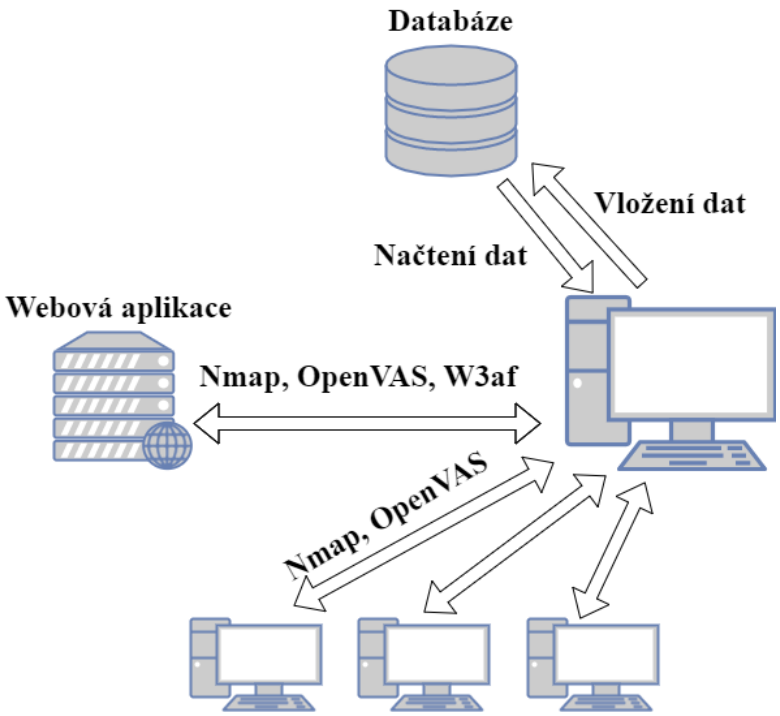
7.4 Výběr komponent

Pro výběr skeneru padla jasná volba na OpenVAS. Podstatnou součástí je i program Nmap. Ten se postará o pravidelné skenování portů. OpenVAS je jako jediný šířen pod licencí GNU/GPL. Všechny skenery vykazovaly velice podobnou dobu skenování. Pokud by výběr byl povolen i na komerční software pak by pravděpodobně přišla řeč na Nessus. Možnosti nastavení jsou poměrně široké, různé druhy skenů a jednoduchý design, jsou jeho výhody. Nexpose našel jednoznačně nejvíce zranitelností a je na bezpečnostním týmu, jak je bude řešit. Problém nastává při orientaci ve webovém rozhraní. Také je, přinejmenším zvláštní, způsob hodnocení zranitelností. Nexpose je ohodnotí na základě CVSS hodnoty a vlastního ohodnocení a nikdo neví, zda je to dobře. Správce by měl sám určit a popř. upravit ohodnocení zranitelností pro jeho firmu.

Dalším článkem je skener webových aplikací. Zde dominuje W3af. Jde o mocný nástroj. Široká škála pluginů a stálá podpora jsou velkou předností. Nikto je na druhou stranou snadno ovladatelný. Bohužel už není oficiálně podporován. Skener bez podpory není dobrý skener. V porovnání s W3af má méně možností nastavení a nenabízí grafické prostředí. Výsledky běhu skeneru Nikto jsou také zpochybnitelné. Komerční webové skenery nebyly testovány.

K vytvoření systému je nutné psát skripty, které budou zajišťovat komunikaci s programy, starat se o nastavení a o práci programů. Jako skriptovací jazyk je zvolen Python 2.7. Vybrán je díky široké nabídce modulů pro práci s jinými programy, nativní podpoře v Ubuntu a jednoduché práci v něm.

Reporty je nutné zanášet do databáze. Vybrána je PostgreSQL databáze. Má velkou základnu nadšenců. Python nabízí moduly pro práci s ní. Je vyvíjena jako open-source. Na databázi bude nasazeno, pro lepší orientaci webové prostředí phpPgAdmin.



Obrázek 22 Diagram

Na diagramu je znázorněn postup, jakým budou procesy probíhat. První sken proběhne programem Nmap na zadaných IP adresách. Data jsou vložena do databáze. Při skenování programem OpenVAS se jednotlivé IP adresy, které jsou uloženy v databázi,

načtou a OpenVAS otestuje výskyt zranitelností. Vygenerovaná data jsou uložena do databáze. Posledním skenem je otestování programem W3af na zadanou webovou aplikaci. Data jsou uložena do databáze.

Některé moduly, které skripty používají, je nutné volat jako správce. Na místě je tedy úprava práv, pravděpodobně přes Access Control List. Vybraný správce systému musí mít, alespoň základní znalosti s prací v linuxové distribuci. Většina modulů do Pythonu je tvořena nadšenci a je tak možné, že se někdy vyskytne chyba.

Zde je seznam používaného softwaru včetně stažených modulů.

Název programu	Verze
Nmap	7.01
OpenVAS	8
W3af	1.7.6
Postfix	3.1.0
PostgreSQL	9.5.6
phpPgAdmin	5.1
Psycopg2	2
Python-nmap	0.6.1
Openvas_lib	1.1.2

Tabulka 1 - Seznam softwaru

7.5 Implementace

Tato kapitola je o samotné implementaci jednotlivých nástrojů a o instalaci dalších potřebných nástrojů. Vše probíhalo na novém systému.

7.5.1 PostgreSQL

Pro vytvoření fungujícího systému, je důležitá instalace databázového systému. Do OS je nainstalován databázový systém PostgreSQL. Ke správné práci v Ubuntu je doinstalován nejen samotný PostgreSQL, ale i balíček závislostí.

```
apt-get install postgresql postgresql-contrib
```

Ještě je nutné upravit heslo správce, v tomto případě je správce postgres. Vše probíhá skrz program psql. Psql je jednoduchý nástroj pro interpretaci příkazů na příkazové řádce. Automaticky je instalován s balíčkem postgresql. Jako první je vytvořena databáze reports. Také je zapotřebí původní databázi postgres smazat, protože působila problémy při

přihlašování. Mezi příkazy pro vytvoření a smazání databáze je nutné se znovu přihlásit, protože není možné smazat databázi ve které aktuálně program běží.

```
sudo -u postgres psql
\password postgres
CREATE DATABASE reports;
sudo -u postgres psql reports
DROP DATABASE postgres;
```

7.5.2 Nmap

Nyní je vytvořena tabulka nmap_report, která obsahuje sloupce id a date, kdy id je jednoznačný identifikátor řádky a automaticky se zvětšuje o jedna a date, který obsahuje datum vytvoření logu.

```
CREATE TABLE nmap_report(
id SERIAL PRIMARY KEY NOT NULL,
date DATE NOT NULL);
```

Podobně je tvořena i tabulka nmap_log, kde budou uloženy informace o jednotlivých portech na adresách. Pro určení souvisejícího reportu, je použit cizí klíč s odkazem na ID reportu.

```
CREATE TABLE nmap_log(
id SERIAL PRIMARY KEY NOT NULL,
host TEXT NOT NULL,
hostname TEXT,
port TEXT NOT NULL,
protocol TEXT NOT NULL,
state TEXT NOT NULL,
name TEXT,
id_report INT REFERENCES nmap_report(id));
```

Databáze se postupem času bude rozrůstat a částečně měnit, protože bude nutné přidávat logy z dalších nástrojů.

V tomto stavu je vhodné vytvořit skript pro ukládání logů z aplikace Nmap do databáze. Skript má za úkol ukládat textové soubory s logy do složky a zároveň do databáze

zanášet údaje. Ve složce uživatele je vytvořena složka nmaplogs, do které se ukládají vytvořené logy pojmenované podle aktuálního dne a dále soubor iplist.txt jehož funkce je vysvětlena později. Také je vytvořena složka scripts, ve které je uložena většina skriptů.

Prvotním krokem je vytvoření souboru nmap_scan.py ve složce scripts.

```
touch nmap_scan.py
```

Přípona .py označuje soubor pythonu. Důležité je nastavit spustitelnost.

```
chmod +x nmap_scan.py
```

Je nutné, aby python podporoval operace s aplikací Nmap. Proto je stažen balíček python-nmap.

```
pip install python-nmap
```

Použitá je poslední stabilní verze 0.6.1. Moduly jsou pomocí importu připraveny k použití. První řádek definuje skriptu, který interpret je nutné použít pro správný chod skriptu. Následují importy modulů využívaných v běhu skriptu. Modul nmap je z výše uvedeného balíčku. OS je zaveden pro specifikování cesty ke složce, kde se budou generovat logy. Psycopg2 je modul pro práci s databází Postgres. Je nutné ho manuálně doinstalovat.

```
pip install psycopg2
```

Datetime je přítomen, protože log bude mít název podle aktuálního dne. Subprocess je modul, který umožňuje zavolat skripty, které jsou v jiné složce. V tomto případě se bude volat skript umístěný ve složce nmaplogs, jenž se bude starat o zjišťování změn v aktuálním logu.

```
#!/usr/bin/python
import nmap
import os
import psycopg2
from datetime import date
import subprocess
```

Funkce Import_log je volána s parametry report_id, datahost, datahostname, dataport, dataprotocol, datastate, dataname. Report_id je použit pro vložení čísla řádky reportu. Ostatní parametry jsou data, která se budou vkládat do tabulky nmap_log. Proměnná sql obsahuje SQL příkaz pro vložení požadovaných údajů do databáze.

```
def Import_log(report_id, datahost, datahostname, dataport,
               dataprotocol, datastate, dataname):
     sql = "INSERT INTO nmap_log (host, hostname, port,
     protocol, state, name, id_report) VALUES(%s, %s, %s, %s,
     %s, %s, %s);"
```

Metodou try-except je zajištěno připojení k databázi a, v případě nepřipojení se, je vypsána chyba.

```
try:
     conn = psycopg2.connect("dbname='reports'
     user='postgres' host='localhost' password=''")
except:
     print "I am unable to connect to the database"
```

Třída cursor dovoluje uživateli provádět v databázové operace. Proto, je vytvořen objekt této třídy, a s metodou execute je schopen zapsat definovaným příkazem předaná data. Následuje potvrzení operace a uzavření komunikace s databází.

```
cur = conn.cursor()
cur.execute(sql, (datahost, datahostname, dataport,
dataprotocol, datastate, dataname, report_id))
conn.commit()
cur.close()
conn.close()
```

Funkce Import_report vypadá obdobně. Rozdíly jsou v parametru a provedení příkazu. Je důležité dát za proměnnou file_name čárku. Bez ní příkaz působí problémy. Funkce vrací hodnotu čísla řádky. Tato hodnota je dále využita v hlavní funkci skriptu.

```
def Import_report(file_name):
     sql = "INSERT INTO nmap_report(date) VALUES(%s)
RETURNING id;"
     …
     cur.execute(sql, (file_name,))
     …
     return row_ID_report
```

Nejdůležitější a hlavní funkcí skriptu je funkce Scanner. V těle této funkce je skryto všechno, co se týče skenování zadaných IP adres. Je zde deklarovaná proměnná file_name. Ta obsahuje aktuální datum ve stringu. Do proměnné report_id je uložena návratová hodnota funkce Import_report s parametrem file_name. Poté je definována cesta, kam ukládat vytvořené reporty. Dále je, díky modulu os, zadána cesta a název reportu do proměnné full_path_name.

```
def Scanner():
     file_name = date.today().strftime('%Y_%m_%d')
     report_id = Import_report(file_name)
     path = '/home/adminvm/nmaplogs'
     full_path_name = os.path.join(path, file_name+".txt")
```

Scanner je instance třídy PortScanner z knihovny nmap. PortScanner nabízí mnoho metod. Právě takovou metodou je scan. Do této metody jsou zadány jako parametry argumenty, s nimiž se Nmap bude spouštět. Přepínače -sU a -sS jsou zde, protože sken proběhne jak na UDP, tak TCP porty. Přepínač -iL udává, kde vzít vytvořený seznam IP adres na kterých proběhne sken. Vytvořený soubor může být ve tvaru jednotlivých IP adres nebo IP adresy a CIDR zápisu. V této řádce se skrývá největší síla aplikace Nmap. Sken je možné ho spouštět s velkým množstvím nastavení, jako je fragmentace paketů, zpoždění skenů, posílání paketů jako počítač v síti, skenování OS a další. Je důležité, aby si správce prostudoval, co který přepínač dělá a podle toho případně modifikoval skript.

```
scanner = nmap.PortScanner()
scanner.scan(arguments = '-sU -sS -iL
/home/adminvm/iplist.txt')
```

Následuje otevření souboru pro zápis. Příkaz with kombinuje příkazy try, except a finally. Tím je umožněno otevřít soubor a zavřít v jednom bloku. Následují cykly for postupně procházející všechny IP adresy, protokoly a porty. Porty jsou seřazeny.

```
with open(full_path_name, 'w') as log_file:
     for host in scanner.all_hosts():
          print '\nHost: %s' % (host) +  " was scanned"
          for protocol in scanner[host].all_protocols():
               port = scanner[host][protocol].keys()
               port.sort()
```

V těle posledního cyklu, jsou postupně plněny daty proměnné, které se jsou vkládány do databáze. Vše je uloženo do proměnné log, která je použita jako argument pro zápis do souboru.

```
for ports in port:
     datahost = "%s" % (host)
     datahostname = "%s" % (scanner[host].hostname())
     …
     log = 'Host: %s\ hostname: %s\ …' % (datahost,
     datahosname, …) + '\n'
```

V tomto cyklu se volá funkce Import_log s potřebnými parametry. Po této definici je proměnná log zapsána do souboru.

```
log_id = Import_log(report_id, datahost, datahostname, …)
log_file.write(log)
```

Tímto končí funkce a v těle skriptu je zavolána funkce Scanner. Po vykonání funkce je volána funkce call z modulu subprocess. Pomocí ní je skript schopen provést skript difference ze složky nmaplogs.

```
Scanner()
subprocess.call("./difference.py", cwd
="/home/adminvm/nmaplogs")
```

Pro pravidelné spouštění je využitý zabudovaný program crontab, která dokáže skript pustit každý den ve zvolenou hodinu. Nastavena je hodnota 10:30, protože lidé již mají spuštěné počítače a případný útočník může mít otevřené porty pro únik dat. Tato úprava zajišťuje nejen spuštění, ale i výpis na otevřený terminál s číslem 10. Případný terminál je možné změnit, jen je nutné zjistit jeho číslo pomocí příkazu tty.

```
crontab -e
30 10 * * * python /home/adminvm/scripts/nmap_scan.py >
/dev/pts/10
```

Skript difference je vytvořen ve složce nmaplogs. Za úkol má vypsat nové řádky v posledním uloženém logu oproti předešlému. V případě zaznamenání změny je poslán email správci. Pro zasílání emailu je nainstalován balíček mailutils.

```
apt install mailutils
```

Při instalaci se zobrazí průvodce instalací. V prvním okně je na výběr, jaká konfigurace mail serveru nejlépe odpovídá požadované funkcionalitě. V tomto případě je vybrán Internet with smarthost, kde se maily posílají pomocí SMTP a samotný mail je zasílán přes vybraného agenta. Tím odpadá výraznější konfigurace mail serveru, ale je nutné upravit soubor main.cf. V dalším okně je nutné zadat doménu, ideálně stejnou, jako má mail server. Nyní je nutné upravit soubor main.cf.

```
gedit /etc/postfix/main.cf
```

V tomto konfiguračním souboru je nutné změnit několik položek. Konkrétně myhostname, myorigin, mydestination a mynetworks.

A nakonec restartovat službu.

```
systemctl restart postfix
```

Ve skriptu difference je nutné importovat některé moduly, a to os pro specifikování cesty a výpis souborů ve složce, modul difflib, kde jsou funkce pro výpis změn, modul smtplib pro zasílání emailů přes protokol SMTP a modul sys pro ukončení skriptu. Posledním importem je import třídy copyfile, která bude využita při zasílání emailu.

```
#!/usr/bin/python
import os
import difflib
import smtplib
import sys
from shutil import copyfile
```

Hlavní funkce Compare, jejíž úkolem je porovnávat změny v souborech, obsahuje proměnnou file_path která, stejně jako v předchozím skriptu, obsahuje cestu k textovému souboru. Podmínka if not zajišťuje, že pokud soubor neexistuje, bude vytvořen. Proměnná log_change je typu boolean a umožňuje určit, zda byla připsána řádka do souboru. Dir_path specifikuje cestu do složky s logy.

```
def Compare():
     file_path = "/home/adminvm/nmaplogs/difference.txt"
     if not os.path.exists(file_path):
          open(file_path,'w').close()
     log_change = False
```

```
    dir_path = "/home/adminvm/nmaplogs"
```

Modul os obsahuje třídu listdir. Ta umožňuje uložit do seznamu jednotlivé soubory ve složce. Poté je použita funkce sort, protože je potřeba soubory seřadit. Seznam je seřazen obráceně, tedy sestupně. Díky tomu je nový log třetí v pořadí a předchozí log jako čtvrtý. První dva soubory jsou skript a vytvořený soubor. Podmínkou je zajištěno, aby logy byly minimálně dva a bylo co porovnávat.

```
list_of_files = os.listdir(path)
list_of_file.sort(reverse=True)
if (len(list_of_files)) < 4:
    print "\nNot enough logs\n"
    sys.exit()
```

Definována je proměnná new_line, kterou je prázdný string. Další krok je otevření logů.

```
new_line = ""
with open(list_of_files[3]) as text1, open(list_of_files[2])
as text2:
```

Proměnná d je instance třídy Differ z modulu difflib. Díky funkci compare, je možné porovnávat řádky a vypsat ty nové. V této chvíli je nutné pouze porovnat řádky a uložit do proměnné jako seznam.

```
d = difflib.Differ()
diff = list(d.compare(text1.readlines(), text2.readlines()))
```

Pro otevření souboru se změnami je použit opět příkaz with. Do proměnné je vložen seznam jako text, díky funkci join.

```
with open(file_path,'w') as diff_file:
    _diff = ''.join(diff)
```

Pokud řádka začíná znakem +, jenž značí novou řádku v novém logu je proměnná log_change nastavena na hodnotu True a řádka je zapsána do souboru. Pomocí proměnné new_line, je vložena prázdná řádka do souboru.

```
for line in _diff.splitlines():
    if line.startswith('+'):
        log_change = True
        diff_line.write(line+'\n')
```

```
            new_line += (line+'\n')
```

Po ukončení for cyklu je vložena podmínka, kde, pokud je hodnota log_change False, je vypsána zpráva a skript ukončen. Pokud je hodnota TRUE, je volána funkce Email.

```
if log_change == False:
     print "\nNothing new\n"
     sys.exit()
else:
     Email(new_line)
```

Následuje definice funkce Email. Funkce má za úkol poslat správci zprávu s případnými novými řádky. Samotná funkce sendmail potřebuje k odeslání emailu 3 parametry – email odesílajícího, přijímacího a samotnou zprávu. Tyto parametry jsou specifikovány jako první.

```
def Email(new_line):
     sender = 'adminvm@tomcat.prf.jcu.cz'
     receiver = ['vnitrni@adresa.cz']
     message = """Subject: New report
     New open ports were found.\n %s""" % new_line
```

Pomocí konstrukce try-except je v těle try, vytvořena instance třídy SMTP z modulu smtplib s parametrem localhost. V případě, že se zaslání nepodaří je vypsána chyba a soubor se změnami je zkopírován do hlavní složky správce, pro případnou pozdější kontrolu. V těle skriptu je volána hlavní funkce, která se vykoná.

```
try:
     smtpObj = smtplib.SMTP('localhost')
     smtpObj.sendmail(sender, receiver, message)
     print("\nNew open ports founded. Sucessfully sent
email\n")
except:
     print("\nError: unable to send mail\n")
     copyfile("difference.txt",
     "/home/adminvm/difference.txt")


Compare()
```

7.5.3 OpenVAS

V tuto chvíli je nutné, aby adresy, které budou uvedeny v databázi logů, byly použity v programu OpenVAS a výsledky vloženy do databáze. Jako první jsou vytvořeny tabulky openvas_log a openvas_report, pomocí programu psql, v databázi. Do tabulky openvas_report jsou zanášeny údaje UUID, což je jednoznačný identifikátor reportu v aplikaci OpenVAS a údaj date jako datum vytvoření.

```
sudo -u postgres psql reports
CREATE TABLE openvas_report(
id SERIAL NOT NULL PRIMARY KEY,
uuid TEXT NOT NULL,
date DATE NOT NULL);
```

Do druhé tabulky jsou vkládány logy z jednotlivých skenů adres. Sloupec report_id obsahuje číslo řádky z předchozí tabulky jako cizí klíč. Důležitými sloupci jsou task, podle které se později v průběhu skriptu, porovnává, zda údaj zapsat či nikoliv, dále nvt, což je jednoznačný identifikátor zranitelnosti, name jako jméno zranitelnosti, description, popis zranitelnost a severity, což znamená, jakou nebezpečnost přiřadil OpenVAS zranitelnosti.

```
CREATE TABLE openvas_result(
id SERIAL NOT NULL PRIMARY KEY,
uuid TEXT NOT NULL,
task INT NOT NULL,
host TEXT NOT NULL,
port TEXT NOT NULL,
nvt TEXT NOT NULL,
name TEXT NOT NULL,
description TEXT NOT NULL,
severity INT NOT NULL,
qod INT NOT NULL,
report_id INT REFERENCES openvas_report(id));
```

Ve složce scripts jsou vytvořeny soubory openvas_scan.py a openvas_database.py. Také jsou jim přidělena spouštěcí práva.

```
touch openvas_scan.py
```

```
touch openvas_database.py
chmod +x openvas_scan.py
chmod +x openvas_database.py
```

Tyto skripty se budou spouštět jednou měsíčně. První z nich, vykonává hlavní funkci skeneru. Pro práci s programem OpenVAS v Pythonu je potřeba nainstalovat balíček openvas_lib.

```
pip install openvas_lib
```

Importovány jsou moduly os a psycopg2, jejichž funkce je popsána už v předešlých skriptech. Z modulů openvas_lib, threading a functools jsou vybrány důležité třídy. U openvas_lib je to VulnscanManager, který se stará o samotné spuštění a konfiguraci skenu a dále VulnscanException, pro identifikaci případné chyby. Z modulu threading je naimportována třída Semaphore, ze které je použita funkce acquire, jejíž úkolem je zamknout vlákno, na kterém probíhá sken. Z modulu functools je naimportována třída partial, která je využitá pro specifikaci argumentů.

```
#!/usr/bin/python
import os
import psycopg2
from openvas_lib import VulnscanManager, VulnscanException
from threading import Semaphore
from functools import partial
```

Globální proměnná counter je použita pro počítání průběhů funkce Scanner.

```
counter = 0
```

První funkce list_of_host má za úkol vybrat IP adresy, z logů programu nmap, za uplynulých 31 dní a seřadit je vzestupně. Poté je potřeba se pokusit připojit k Postgres databázi. Postup bude stejný jako u první skriptů.

```
def list_of_host():
     sql = "SELECT DISTINCT host FROM nmap_log INNER JOIN
     nmap_report ON nmap_log.id = nmap_report.id WHERE
     nmap_report.date > current_date - INTERVAL '31' day
     ORDER BY host ASC;"
     try:
```

```
          conn = psycopg2.conner("dbname = 'reports' user =
          'postgres' host = 'localhost' password = ''")
     except:
          print("I am unable to connect to the database")
```

Dále je definován objekt, pomocí kterého je možné se pohybovat po databázi. Vykonán je příkaz sql. Pro zachycení řádků je použita funkce fetchall a je uložena do proměnné IP_list. Následuje potvrzení příkazu a uzavření komunikace s databází. Jako návratová hodnota je vracen IP_list.

```
cur = conn.cursor()
cur.execute(sql)
IP_list = cur.fetchall()
conn.commit
cur.close()
conn.close()
return IP_list
```

Funkce status bude obstarávat pouze výpis průběhu skenu v procentech.

```
def status(i):
     print(str(i)+ '%')
```

Funkce Scanner plní hlavní funkci skriptu. První je volána globální proměnná counter. Do proměnné sem je uložena funkce Semaphore s hodnotou 0. Tato funkce se stará o synchronizaci vláken a umožňuje, tedy pustit pouze jeden sken a až po jeho doběhnutí je možné spustit další. Po uvolnění se funkcí release nastaví hodnota na 1, ale funkce acquire hodnotu opět sníží na 0 a vlákno se opět uzamkne. Do proměnné IP_list je uložena návratová hodnota funkce list_of_host.

```
def Scanner():
     global counter
     sem = Semaphore(0)
     IP_list = list_of_host()
```

Metoda fetchall uložila nalezené výsledky do typu tuple. Tento typ je podobný jako seznam, ale objekty uložené v tuple se nedají měnit. Definován je kulatými závorkami – tup = (). Pro

vybrání objektů v něm je třeba použít dvojitý for cyklus, kde v prvním projdeme objekty tuple a v druhém už je možné načíst uložené hodnoty.

```
for tuple in IP_list:
     for IP_string in tuple:
```

Následující operace probíhají ve druhém for cyklu. Jako první je do proměnné scanner uložena třída VulnscanManager s parametry localhost nebo adresy na které OpenVAS pracuje, uživatelského jména a hesla.

```
scanner = VulnscanManager("localhost", "admin", "admin")
```

Na začátku skenování je vypsán výpis se skenovanou IP adresou. Dále je definována samotná konfigurace OpenVAS skenu. Do target_id je zavedena metoda launch_scan s argumenty. IP adresa je uvedena první, poté se definuje profil skenu jako je Full and Fast, Full and very deep a další. Navazuje callback_end, což je parametr, který pokud je definován, dovoluje spustit sken v pozadí a na konci skenování je zavolána funkce, která je definována. V tomto případě se zavolá funkce lambda, která podporuje vytváření anonymních funkcí. Uvolní vlákno a zavolá se Semaphore, který bude nastaven na 1. Posledním argumentem je callback_progress, který každých 10 vteřin vypíše do terminálu průběh. Sem.acquire pak znovu zamyká vlákno. Na závěr je vypsáno oznámení o konci skenu a hodnota čítače se zvýší o 1. Poté už je v těle skriptu volána metoda Scanner.

```
     print „Scan progress for IP: %s“ % string
     scan_id, target_id = scanner.launch_scan(target =
IP_string,
     profile = „Full and Fast“, callback_end = partial(lambda
     x: x.release(), sem), callback_progress = status)
     sem.acquire()
     print „Scan is done for %s“ % IP_string
     counter +=1
Scanner()
```

Nyní je nutné vložit výsledky do vytvořených tabulek v databázi. Pro nový skript je nutné nejen importovat moduly pro práci s Postgres databázemi, ale i SQLite3, protože samotná databáze programu OpenVAS je právě v tomto systému. Data je tedy nutné nejprve vybrat z SQLite databáze, uložit a vložit do Postgres databáze. Toto je řešeno ve skriptu

openvas_database. Mezi prvními importovanými moduly jsou sqlite3 a psycopg2 pro práci s databázemi, dále openvas_scan. Ten se bude spouštět, díky importu, jako první a není tak nutné řešit, volání modulem subprocess jako v předchozích skriptech. Posledním volaným modulem je datetime, který bude využit pro specifikaci datumu. Globální proměnná task_report je definována nulu. Proměnná bude použita jako rozhodovací kritérium ve funkci sqlite_openvas_result.

```
#!/usr/bin/python
import sqlite3
import psycopg2
import openvas_scan
from datetime import date
task_report = 0
```

Popis funkcí skriptu nebude řazen tak, jak jsou funkce napsány, ale jak jdou za sebou v průběhu skriptu. Na samotném začátku je volán importem, skript openvas_scan, který je popsán výše. V hlavním těle cyklu je volána funkce sqlite_openvas_report, jenž se stará o vybrání záznamů z SQLite databáze. Na počátku se deklaruje globální proměnná task_report a lokální uuid_report. Připojení k databázi SQLite, obstarává podobný příkaz jako u Postgres, s tím rozdílem, že zde se připojuje příkazem přímo k souboru s příponou db. Soubor tasks.db je databáze tvořená programem OpenVAS. Třída cursor pak obstarává provádění operací v databázi.

```
global task_report
uuid_report = ""
conn = sqlite3.connect('/var/lib/openvas/mgr/tasks.db')
```

Pro vybrání správných uuid a task z tabulky reports je nutné vybírat jen ty s id, která jsou platná pro předchozí měsíc. To je již ošetřeno v předchozím skriptu, ale do tohoto skriptu je nutné přenést čítač používaný v něm a podle něho vybrat ty nejaktuálnější.

```
cur = conn.cursor()
cur.execute("SELECT uuid, task FROM reports WHERE id IN
(SELECT id FROM reports ORDER BY id DESC LIMIT %s);" %
openvas_scan.counter)
```

Nyní jsou proměnným přiřazovány jednotlivé hodnoty a na konci tohoto cyklu je volána metoda Openvas_report s parametrem uuid_report, který je zanášen do Postgres databáze.

Do proměnné task_report je také přiřazováno, ale není zanášena do databáze. Je využita až v pozdějším běhu skriptu. Následuje potvrzení operace a ukončení komunikace s databází.

```
for row in cur:
     uuid_report = row[0]
     task_report = row[1]
     Openvas_report(uuid_report)
conn.commit()
cur.close()
conn.close()
```

V tuto chvíli se běh skriptu přesouvá do funkce Openvas_report. Zde je nastavena proměnná file_name jako datum. Do sql_report je vložen příkaz pro vložení dat do Postgres databáze a bude vracet hodnotu id řádky. Následuje pokus o připojení se k databázi a pokud je připojení úspěšné, deklaruje se cur jako kurzor.

```
def Openvas_report(uuid_report):
     file_name = date.today().strftime('%Y_%m_%d')
     sql_report = "INSERT INTO openvas_report (uuid, date)
     VALUES(%s,%s) RETURNING id;"
     try:
          conn = psycopg2.connect("dbname='reports'
          user='postgres' host='localhost' password='admin')
     except:
          print("I am unable to connect to the database")
     cur = conn.cursor()
```

Vykonání příkazu pro vložení s hodnotami uuid_report a file_name, obstará příkaz cur.execute. Obstarání čísla řádky, která bude využita pro přiřazení cizího klíče, má na starosti funkce fetchall. ID řádky je vloženo do proměnné row_ID_report. Následuje potvrzení vložení.

```
cur.execute(sql_report, (uuid_report, file_name))
row_ID_report = cur.fetchall
conn.commit()
```

ID řádky je, kvůli funkci fetchall, opět zabalen do typu tuple. K samotné hodnotě je možné se dostat přes dvojitý for cyklus. Ve druhém for cyklu je volána funce sqlite_openvas_result s číslem řádky. Poté je uzavřena komunikace s databází.

```
for tuple in row_ID_report:
     for index_row in tuple:
          sqlite_openvas_result(index_row)
cur.close()
conn.close()
```

Ve funkci sqlite_openvas_result jsou jako první definovány proměnné. Zde se také volá globální proměnná task_report s uloženou hodnotou. Následuje navázání komunikace s SQLite databází. Funkce cursor je vysvětlena výše. Dalším krokem je vytvoření SQL dotazu. V něm je vybrány hodnoty potřebné pro zanesení do vytvořené uživatelské databáze. Tabulka results je přes inner join propojena s tabulkou nvt, kde jsou obsaženy informace o zranitelnostech. Ve vnitřním dotazu je potřeba vybrat reporty seřadit top x, kde x je hodnota čítače z předchozího skriptu.

```
uuid_result = “”
task_result = 0
…
global task_report
try:
     conn = sqlite3.connect('/var/lib/openvas/mgr/tasks.db')
except:
     print “I am unable to connect to the database”
cur = conn.cursor()
cur.execute(“SELECT results.uuid, results.task, host, port,
nvt, name, description, severity, results.qod FROM results
INNER JOIN nvts ON results.nvt = nvts.uuid WHERE report IN
(SELECT DISTINCT report FROM results ORDER BY report DESC
LIMIT %s);” % openvas_scan.counter)
```

Pro každou, takto vybranou řádku, je nutné porovnat, zda se vybraná task_result rovná, již dříve definované, hodnotě uložené v task_report. Pokud ano, pak je možné zanést vybrané údaje do databáze. To je umožněno metodou Openvas_result s parametry index_row a vybraných položek z databáze. Opět následuje potvrzení a ukončení komunikace s databází.

```
for row in cur:
     task_result = row[1]
     if task_report == task_result:
          uuid_result = row[0]
          …
          Openvas_result(index_row, uuid_result, …)
conn.commit()
cur.close()
conn.close()
```

Poslední používanou funkcí je funkce Openvas_result, která je vykonávána s parametremy index_row, uuid_result a dalšími. Metoda je podobná první funkci Openvas_report, pouze s rozdílnými daty, která jsou vkládána do databáze.

```
def Openvas_result(index_row, uuid_result, …):
     sql = "INSERT INTO openvas_result(uuid, task, host,
port, nvt, name, description, report_id, severity, qod)
     VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s);"
     try:
     …
     except:
     …
     cur = conn.cursor()
     cur.execute(sql, (uuid_result, task_result, …))
     conn.commit()
     cur.close()
     conn.close()
```

Tímto jsou dokončené operace s nástrojem OpenVAS.

Pro lepší orientaci, je nainstalován phpPgAdmin, webový administrátor databáze, a v něm jsou vytvořeny pohledy. Je zapotřebí nastavit extra_login_security na false, pokud se uživatel chce přihlásit jako uživatel postgres.

```
apt-get install phppgadmin
nano /etc/phppgadmin/config.inc.php
$conf['extra_login_security'] = false;
```

Administrátor nabízí jednoduchý přehled databází a tabulek. Poskytuje i možnost tvorby pohledů. Pohled je SQL dotaz, jehož výsledkem je tabulka s požadovanými daty. Jako první je vytvořen dotaz, který vybere IP adresy se zranitelností, které mají vážnost vyšší nebo rovnou 7. Zároveň je vypsán sloupec s označením zranitelnosti a jejím jménem.

```
SELECT openvas_result.host, openvas_result.nvt,
openvas_result.name, openvas_result.severity
FROM openvas_result
WHERE openvas_result.severity >=7;
```

Obdobně je vytvořen pohled s vážností větší než 5. Další pohled je vytvořen pro nalezení IP adresy podle jména zranitelnosti. Jméno lze měnit pomocí tlačítka Alter u položky Definition.

```
SELECT openvas_result.host, openvas_result.name,
openvas_result.nvt, openvas_result.description FROM
openvas_result WHERE openvas_result.name = 'Traceroute';
```

Vytvořené pohledy je možné snadno rozšířit v záložce Views. Pravidelného spouštění je docíleno díky programu crontab. Pouštět se bude každý 15. den v měsíci ve 14:30. Vypisovat se bude na stejný shell ,jako na ten s programem Nmap. Samozřejmě je možná změna.

```
crontab -e
30 14 15 * * python /home/adminvm/scripts/openvas_database.py
> /dev/pts/10
```

7.5.4 W3af

Posledním potřebným programem je W3af. Jeho hlavní funkcí je otestovat zadané webové stránky na případné zranitelnosti. Ve složce w3af je vytvořen soubor web_sql.w3af. Úkolem tohoto souboru je spustit se jako konfigurační soubor pro W3af a ten pomocí něj získá požadovanou konfiguraci. Úkolem testu je zjistit zranitelnosti typu sql injection, xss a os command.

Do souboru typu w3af se zapisují příkazy, jak jdou, v programu W3af, za sebou. Jako první je nastavení složky profiles, ze které je použit empty_profile, protože vše bude v průběhu nastaveno. Nesmí být opomenut na příkaz back, který konfiguraci ukládá po

každém kroku. Ve složce plugins, je zapnuta volba pro mapování stránek, ale pouze těch, které se pojí se zadaným cílem. Následuje definování hlavních testovacích metod, jako je blind_sqli, os_commanding, sqli a xss. Za výstupní formát je zvolen csv. Konfigurována je i cesta do výstupní složky. Posledními kroky jsou zadání webu a vyčištění od přechozího testu. Sken může začít.

```
profiles
use empty_profile
back
plugins
crawl config web_spider
set only_forward TRUE
back
audit blind_sqli os_commanding sqli xss
output csv_file
output config csv_file
set output_file /home/adminvm/w3aflogs/w3af.csv
back
back
target
set target http://zvoleny web
back
cleanup
start
```

V této formě je možné skenovat webové stránky na zadané zranitelnosti. Pro ovládání a import do databáze je ve složce scripts vytvořen skript w3af_script. První je opět řádka o používaném interpretu a další o importovaných modulech. Mezi ty známé přibyl copyfile, protože je nutné vytvořený csv soubor nakopírovat do složky /tmp/. V původní složce je problém s přístupovými právy. Druhým novým importem je csv. Zahrnut je z důvodu práce se souborem csv.

```
#!/usr/bin/python
import subprocess
import sys
```

```
import psycopg2
import os
from datetime import date
from shutil import copyfile
import csv
```

První funkcí je csv_file. Tato funkce se stará o otevření csv souboru, načtení záznamů ze sloupců a předání uložených dat funkci w3af_log. Při otevírání je nutné specifikovat delimiter, česky oddělovač. Také je nutné specifikovat quotechar, který odděluje začátek a konec záznamů. Pro správnou práci bylo nutné upravit samotný w3af skript, který se staral o výstup do csv souboru. Působil potíže, protože jako oddělovač byla použita čárka. Čárka je v seznamu používaná pro oddělování jednotlivých položek. Čárka byla nahrazena středníkem.

```
def csv_file(row_id):
     with open('/tmp/w3af.csv','rb') as csvfile:
          reader = csv.reader(csvfile,
          delimiter=';', quotechar'|')
          for data in reader:
               severity = data[0]
               name = data[1]
               …
               w3af_log(severity, name, …, row_id)
```

Následující funkce, se stará o vkládání záznamu o reportu, do databáze. Importovat se bude pouze datum. Návratová hodnota je id řádky. Kód vypadá obdobně jako v ostatních kódech s databází.

```
def w3af_report(file_name):
     sql = "INSERT INTO w3af_report (date) VALUES(%s)
     RETURNING id;"
     try:
          …
     except:
          …
     cur = conn.cursor()
```

```
    cur.execute(sql, (file_name,))
    row_ID_report = cur.fetchone()[0]
    conn.commit()
    cur.close()
    conn.close()
    return row_ID_report
```

Další funkce, w3af_log, se stará o vkládání logů do tabulky. Kód je velice podobný jako u vkládání logů z programu Nmap. Liší se pouze vkládané hodnoty. Ty jsou předávány parametrem.

```
def w3af_log(severity, name, …, row_ID):
    sql = "INSERT INTO w3af_log (severity, name, row_ID, …)
    VALUES(%s,%,%s, …)"
    try:
        …
    except:
        …
    cur = conn.cursor()
    cur.execute(sql,(severity, name, …, row_ID))
    conn.commit()
    cur.close()
    conn.close()
```

Poslední a hlavní funkce má za úkol zavolat skript s parametrem -s, který dovoluje programu W3af spustit test s konfiguračním souborem. Toho je docíleno funkcí call z modulu subprocess a funkcí executable z modulu sys. Na navazujícím řádku je specifikováno jméno. Tím je datum. Do proměnné row_ID je vložena návratová hodnota funkce w3af_report. Díky příkazu copyfile je možné překopírovat soubor do adresáře tmp. Důvod je uveden výše. Dále je volána funkce csv_file s parametrem row_ID. Poté je funkcí remove z modulu os smazán dočasný soubor v adresáři tmp. Pro nové jméno je nastavena proměnná new_name, která vezme proměnou file_name a připojí příponu csv. Vše musí být ve stringu. Nakonec dojde funkcí rename k samotnému přejmenování. Nezbývá než v hlavním těle skriptu, volat funkci w3af.

```
def w3af():
```

```
    subprocess.call([sys.executable,
    '/home/adminvm/w3af/w3af_console','-
    s','/home/adminvm/w3af/web_sql.w3af'])
    file_name = date.today().strftime('%Y_%m_%d')
    row_ID = w3af_report(file_name)
    copyfile("/home/adminvm/w3aflog/w3af.csv",
    "/tmp/w3af.csv")
    csv_file(row_ID)
    os.remove("/tmp/w3af.csv")
    new_name = str(file_name+'.csv')
    os.rename("/home/adminvm/w3aflogs/w3af.csv",
    "/home/adminvm/w3aflogs/"+new_name)
w3af()
```

Ještě je potřeba vytvořit příslušné tabulky v databázi. První tabulka obsahuje pouze záznam o vytvořeném reportu.

```
sudo -u postgres psql reports
CREATE TABLE w3af_report(
id SERIAL PRIMARY KEY,
date DATE NOT NULL);
```

Druhá tabulka pak obsahuje jednotlivé zranitelnosti z reportu.

```
CREATE TABLE w3af_log(
id SERIAL PRIMARY KEY,
severity TEXT NOT NULL,
name TEXT NOT NULL,
method TEXT NOT NULL,
uri TEXT,
token TEXT,
base64 TEXT,
vulnerability_id TEXT NOT NULL,
description TEXT NOT NULL,
id_report INT REFERENCES w3af_report(id));
```

8 Závěr

V bakalářské práci se mi podařilo vybrat definice pojmu Vulnerability Management a upravit je pro účely této práce. Každý zdroj se na Vulnerability Management dívá z trochu jiného pohledu. V práci jsem se zabýval první dvěma kroky, které souvisí především s hledáním a posuzováním zranitelnost.

Jsou zde vytyčeny rozdíly mezi penetračním testováním a Vulnerability Managementem. Pro mnoho lidí je definice obou pojmů nejasná a neví co znamenají. V práci je jasně definováno, co který pojem znamená. Lidé by měli být obezřetní při používání obou pojmů a přestat je nesprávně zaměňovat.

Dále je v práci vybráno několik nástrojů pro hledání a posuzování zranitelností. Z výsledků vychází, že volně dostupné nástroje se dají srovnávat s těmi komerčními. Problémem se pak stává cena, kdy za údržbu komerčního systému, jsou společnosti nuceny dávat poměrně velké výdaje. Volně dostupné nástroje mohou mít ten problém, že se o ně komunita přestane starat, jako se stalo s program Nikto.

V praktické části se mi pak podařilo sestavit systém, který je vhodný pro použití ve Vulnerability Managementu. Je zajištěno pravidelné skenování zadané sítě a případných webových aplikací. Veškeré logy jsou ukládány do Postgres databáze. Pro grafické znázornění je využit phpPgAdmin, který nabízí možnost tvoření pohledů.

Práci je možné rozšířit několika způsoby. Práce pokrývá první dva kroky celého procesu Vulnerability Managementu. Zde se nabízí práce, která by dokázala spojit vytvořený systém a systém, který se bude starat o hledání a aplikaci záplat. Další možnou prací je vytvořit grafické rozhraní pro systém. Zajímavým navazujícím projektem by bylo, chopit se Vulnerability Managementu z procesní stránky a optimalizovat jeho běh ve firmě.

Práce bude využita pro Ústav aplikované informatiky a doufám, že splní očekávání. V navazujícím studiu pak přemýšlím o dalších rozšířeních. Například právě grafické rozhraní a správa uživatelů systému.

Seznam literatury

[1] *Open Source Initiative* [online]. b.r. [cit. 2017-04-14]. Dostupné z: https://opensource.org/

[2] ZENG, Yuanyuan, David COFFEY a John VIEGA. How Vulnerable Are Unprotected Machines on the Internet?. In: FALOUTSOS, Michalis a Aleksandar KUZMANOVIC. *Passive and Active Measurement* [online]. Springer International Publishing, 2014, s. 224 [cit. 2017-02-16]. DOI: 10.1007/978-3-319-04918-2_22. ISBN 978-3-319-04918-2. ISSN 0302-9743. Dostupné z: http://link.springer.com/10.1007/978-3-319-04918-2_22

[3] KANDEK, Wolfgang. *Vulnerability Management For Dummies* [online]. 2. West Sussex: John Wiley & Sons, Ltd, 2015 [cit. 2016-12-03]. ISBN 978-1-119-05829-8. Dostupné z: https://www.qualys.com/docs/vm-for-dummies-2nd-edition.pdf

[4] CHUVAKIN, Anton. Vulnerability and Security Configuration Assessment Solutions Comparison. In: *Gartner* [online]. 2012 [cit. 2017-02-08]. G00225382. Dostupné z: http://www.gartner.com/technology/media-products/reprints/qualys/Qualys_225382.html

[5] MELL, Peter, Tiffany BERGERON a David HENNING. SPECIAL PUBLICATION 800-40. *Creating a Patch and Vulnerability Management Program*. Version 2.0. Gaithersburg: National Institute of Standards and Technology, 2005.

[6] SAGER, Tony. *The Cyber OODA Loop: How Your Attacker Should Help You Design Your Defense* [online]. 2015 [cit. 2017-02-01]. Dostupné z: http://csrc.nist.gov/news_events/cif_2015/security-automation/day3_security-automation_930-1020.pdf

[7] GEORGE, Torsten. The Truth About Penetration Testing Vs. Vulnerability Assessments. In: *Security week* [online]. 2016 [cit. 2017-02-08]. Dostupné z: http://www.securityweek.com/truth-about-penetration-testing-vs-vulnerability-assessments

[8] Vulnerability Assessments Versus Penetration Tests. In: *SecureWorks* [online]. 2015 [cit. 2017-04-09]. Dostupné z: https://www.secureworks.com/blog/vulnerability-assessments-versus-penetration-tests

[9] *Common Vulnerability Scoring System v3.0: Specification Document: Specification Document*. Verze 1.7. 2014.

[10] About CVE. *Common Vulnerabilities and Exposures* [online]. 2016 [cit. 2017-02-08]. Dostupné z: https://cve.mitre.org/about/

[11] Top 50 Products By Total Number Of "Distinct" Vulnerabilities in 2016. *CVE Details* [online]. 2017 [cit. 2017-02-01]. Dostupné z: http://www.cvedetails.com/top-50-products.php?year=2016

[12] NVD Frequently Asked Questions. *National Vulnerability Database* [online]. 2017 [cit. 2017-02-08]. Dostupné z: https://nvd.nist.gov/general/faq

[13] Discovery. *CERT* [online]. 2017 [cit. 2017-02-01]. Dostupné z: http://www.cert.org/vulnerability-analysis/research/discovery.cfm

[14] About The Exploit Database. *Exploit Database* [online]. 2017 [cit. 2017-02-08]. Dostupné z: https://www.exploit-db.com/about/

[15] *Nexpose* [online]. b.r. [cit. 2017-04-14]. Dostupné z: https://www.rapid7.com/products/nexpose/

[16] *Nessus Vulnerability Scanner* [online]. b.r. [cit. 2017-04-14]. Dostupné z: https://www.tenable.com/products/nessus-vulnerability-scanner

[17] *Qualys FreeScan* [online]. b.r. [cit. 2017-04-14]. Dostupné z: https://www.qualys.com/forms/freescan/

[18] *OpenVAS* [online]. 2008 [cit. 2017-02-28]. Dostupné z: http://www.openvas.org/about.html

[19] Qualys Introduces Express Lite for Small Businesses. In: *Qualys* [online]. Redwood City, 2013 [cit. 2017-04-07]. Dostupné z: https://www.qualys.com/company/newsroom/news-releases/uk/2013-06-10-qualys-introduces-express-lite-for-small-businesses/

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1 - Skripty

Příloha 2 - Příručka