elektrorevue
ISSN 1213 - 1539

Rok / Year:	Svazek / Volume:	Číslo / Number:	Jazyk / Language
2021	23	2	CZ

Návrh zpracování datasetu u metod strojového učení pro odhalování útoků v oblasti informační bezpečnosti

Dataset processing design in machine learning methods for information security attack detection

Tomáš Slunský

xsluns01@stud.feec.vutbr.cz

Fakulta elektrotechniky a komunikačních technologií VUT v Brně

DOI: -

Abstract: The article analyzes current vulnerabilities of web applications and how to effectively prevent them using machine learning methods. The article also reflects on the current events through a vulnerability that appeared at the end of 2020 and negatively affected a number of websites in the Czech Republic. The aim of the paper will be to demonstrate the specific vulnerability, analyse its potential impact and propose improvements to existing methods for detecting attacks using machine learning techniques. The proposed solution is based on knowledge of web applications and frameworks implemented in PHP (Hypertext Preprocessor).

elektrorevue
ISSN 1213 - 1539

Návrh zpracování datasetu u metod strojového učení pro odhalování útoků v oblasti informační bezpečnosti

Tomáš Slunský

Fakulta elektrotechniky a komunikačních technologií VUT v Brně
Email: xsluns01@stud.feec.vutbr.cz

Abstrakt *– Článek analyzuje současné zranitelnosti webových aplikací a možnosti jak jím efektivně předcházet s využitím metod strojového učení. Článek rovněž reflektuje současné dění skrze zranitelnost, která se objevila koncem roku 2020 a negativně ovlivnila celou řadu webů v České Republice. Cílem článku bude konkrétní zranitelnost demonstrovat, analyzovat její možné dopady a navrhnout vylepšení stávajících metod pro detekci útoků s využitím technik strojového učení. Navržené řešení bude vycházet ze znalosti webových aplikací a frameworků implementovaných v jazyce PHP (Skriptovací programovací jazyk – Hypertext Preprocessor).*

1 Úvod

V současnosti význam WWW (Celosvětová síť – *World Wide Web*), webových aplikací a obecně technologií spjatých s jejich vývojem neustále roste. Hlavním důvodem je stále fakt, že informace z webu jsou snadno dostupné odkudkoli na světě. Dalším zcela zjevným důvodem růstu byla jednoduchost, jak lze dnešní webové aplikace nasazovat. Nelze také opomenout komunikační proces poskytovaný protokolem HTTP (Protokol pro přenos hypertextových dokumentů – *Hypertext Transfer Protocol*) a dalšími s ním souvisejícími, které rozvoji pomohly.

Souhrou těchto okolností vznikla celá řada e-commerce společností a na tomto principu vybudovaly celé své podnikání. Ke změnám dochází také v případě inovací, kdy společnosti integrují webové technologie od průmyslu po bankovnictví atd. S rostoucími požadavky na webové aplikace bylo nutné zavést nové technologie, díky nimž bylo možné lépe komunikovat s uživatelem a nabídnout pokročilejší služby a funkce. Od začátku internetu se webové prezentace posunuly významně vpřed a s nimi i spojené požadavky na údržbu a vývoj. Se zaváděním nových technologií a růstem požadavků jsou spojeny také bezpečnostní problémy a zranitelná místa.

Bezpečnost webových aplikací se stává stále důležitějším tématem. Z tohoto důvodu byl za účelem zlepšení zabezpečení založen OWASP (Projekt zabývající se bezpečností webových aplikací – *Open Web Application Security Project*)[1]. OWASP také, v rámci jednoho z projektů, udržuje seznam 10 aktuálně nejkritičtějších zranitelností webových aplikací, které je nutné uvažovat při návrhu, vývoji a provozu webové aplikace. V současnosti více jak 75 % všech aplikací na webu běží na PHP (Skriptovací programovací jazyk – *Hypertext Preprocessor*) a je tedy nejpoužívanějším programovacím jazykem na straně serveru pro účely webových aplikací. Díky této oblibě PHP vznikla celá řada frameworků, které usnadňují vývoj takových aplikací [18]. Jednotlivé frameworky mají kolem sebe velké komunity vývojařů, které se starají o jejich údržbu. Výhodou webových frameworků je ten, že slouží jako skvělý startovní bod nové webové aplikace, nabízí řadu užitečných funkcí či knihoven, které jsou potřebné již v konkrétních implementacích. Každý framework již implementuje bezpečnostní prvky, na kterých pak taková webová aplikace staví. Příkladem takových frameworků v PHP jsou např. Nette, Laravel, CodeIgniter, Symfony a další.

Z hlediska bezpečnosti může být koncept vývoje velkými komunitami velice zranitelný z důvodu otevřených kódů. Především se jedná o případy, kdy je vyvíjena již konkrétní aplikace komunitně. Pro útočníka pak již není tak složité napsat útok na konkrétní část aplikace nebo subsystém. Dalším problémem je využívání knihoven třetích stran. Jedná se o případy, kdy konkrétní aplikace napsaná nad určitým frameworkem není závislá pouze na bezpečnosti frameworku samotného, ale také na bezpečnosti balíčků či knihoven třetích stran, které framework využívá. Problém je o to větší, pokud se aplikace spoléhá na řešení zabezpečení právě na úrovní takových knihoven.

Tento článek se zabývá souhrnem současného stavu v oblasti bezpečnosti webových aplikací. Hlavní náplní článku je demonstrace zranitelnosti, která se objevila koncem roku 2020 a negativně ovlivnila velkou část českého internetu. Tato zranitelnost bude simulována a též bude demonstrován její dopad v případě zneužití. Na závěr bude navrženo efektivní řešení pomocí metod strojového učení.

2 Současný stav zranitelností webových aplikací a oblastí pro jejich detekci

V současnosti existuje celá řada hrozeb a zranitelností[2] webových aplikací, které potencionální útočníci mohou

[1] https://owasp.org/

[2] Zranitelností webu je v této práci míněna slabina kódu nebo nesprávná konfigurace webu či webové aplikace, která umožňuje útočníkovi získat určitou úroveň kontroly nad webem a v tom nejhorším případě nad celým serverem.

elektrorevue
ISSN 1213 - 1539

zneužít. Ve většině případech se jedná o chyby způsobené vývojem konkrétní webové aplikace nebo frameworku, na kterém běží. Existují ale i jiné zranitelnosti, které s konkrétní implementací aplikace již nesouvisí a útoky jsou vedené odlišným způsobem. Může se jednat např. o útok DDoS (Distribuované odepření služby – *Distributed denial of service*), ke kterému není nutné znát konkrétní implementaci webové aplikace.

V případě zaměření se na zranitelnosti webových aplikací tedy jde především o chyby v implementaci samotných frameworků nebo konkrétních aplikací. Jedním z nejkritičtějších útoků jsou typy injekcí. Injekce samy o sobe jsou také prvním na seznamu projektu OWASP Top Ten, které patří mezi nejvíce zraňující typy útoků současnosti. Další velice rozšířený typ útoků je CSRF (Podvrhnutí požadavku – *Cross-site request forgery*). CSRF sice v současnosti na seznamu OWASP nefiguruje, ale je stále velice zneužívány. Jedná se o útok, kdy dojde k vykonání konkrétní akce na vzdálené webové stránce[3] bez vědomí uživatele. Zajímavé na útoku je, že útok je vykonán např. prohlížením uživatele na webové aplikace A, ale finální útok se děje právě na vzdálené webové aplikaci B bez nutnosti jakékoliv další interakce s webovým serverem A.

Jednoduchým příkladem může být útok přes načítání obrázku na závadné stránce. Útočník do atributu **src**[4] vloží adresu serveru B[5] s konkrétním **GET** požadavkem. V momentě, kdy oběť vstoupí na stránku A, začne docházet k překladu HTML kódu a k sestavování DOM (Objektový model dokumentu – *Document Object Model*). V tomto kroku dojde k načítání smyšleného obrázku a k vykonání požadavku na serveru B, kde dojde k provedení určité akce např. odhlášení ze serveru, odeslání formuláře v momentě, kdy je oběť na serveru přihlášena apod.

2.1 Automatická analýza logů

V případech automatické analýzy dochází ke zkoumání a prohledávání shromažďovaných dat. Může být prováděna v reálném čase (real-time analýza) nebo již na dříve uložených datech. První zmíněná možnost se hodí v situacích pro odhalování kritických stavů. V těchto případech je pak vyvolána příslušná procedura tj. poplach, email apod. Naproti tomu analýza na starších záznamech se využívá spíše pro získání statistických dat. Nejproblematičtější je případ zjištění neznámých nepříznivých událostí, ke které se využívá detekce anomálií[13].

Automatická analýza je vhodná a využívaná v případě velkých informačních systémů a je nutné, aby běžela nad standardizovanými nebo předpřipravenými logy s určitým formátem. Obvykle je tento proces realizován pomocí normalizace, které předchází agregace logů na centrální místo a jejich následná filtrace. Posledním krokem automatické analýzy záznamů je korelace. Jedná se o nejvíce problematickou část na realizaci. Korelace na základě podobných či zcela rozdílných událostí vyvozuje potřebnou znalost z hlediska bezpečnosti, o které bude informovat. Lze ji vytvářet na základě modelu či pravidel.[13]

2.1.1 Detekce incidentů pomocí signatur

Detekce kybernetických incidentů založená na signaturách disponuje nízkou hladinou falešně pozitivních hlášení. Funguje na bázi pravidel, které porovnávají síťový provoz se svojí databází tj. otisků známých útoků a hrozeb. V momentě kdy dojde k pozitivní detekci je spuštěna příslušná rutina např. alarm apod. [6] Nevýhodou tohoto přístupu je, že dokáže rozpoznat pouze útoky, které eviduje ve své databázi a např. na nové hrozby již není účinná.

Metody založené na odhalování hrozeb pomocí signatur mohou být implementovány s odlišnými rozhodovacími pravidly a je možné je dle jejich implementace pravidel rozdělit na modelování na základě stavu, expertní systém a porovnávání řetězců.

2.1.2 Detekce incidentů pomocí anomálií

Detekce kybernetických incidentů pomocí anomálií je zcela odlišný přístup od výše zmíněných signatur. Tato detekční metoda definuje normální chování pomocí předem definovaných vztahů, aby byla poté schopna odlišit odchylky od tohoto normálního chování. V případě, že bude překročena prahová hodnota, která definuje maximální možnou odchylku od normálu, bude tento jev označen za anomálii.[4] Výhodou toho řešení je, že detekce pomocí anomálií dokáže zareagovat i na neznámé hrozby. Nevýhodou řešení je, že vzhledem k obecnému návrhu může docházet k falešně pozitivnímu hlášení.

2.2 IPS a IDS systémy

Jedná se o softwarové nebo hardwarové řešení, které monitoruje síťový provoz a automaticky detekuje podezřelou aktivitu. Hlavní činnosti systému IDS jsou detekce narušení, která by mohla vést k prolomení bezpečnosti. Systémy IDS nemohou zablokovat možný pokus o vniknutí, což je rozdíl vůči systémům IPS. Na rozdíl od pasivního IDS je IPS schopen aktivně zabránit síťovým útokům zahazováním paketů z podezřelých zdrojů a také je schopen zcela zabránit připojení ze zdrojové IP adresy [9].

Systémy využívají pro odhalování kybernetických incidentů detekci pomocí anomálií nebo rozpoznáváním signatur. V případě detekce signatur je potřeba udržovat databázi otisků jednotlivých útoků resp. řetězců specifických pro daný typ útoku. Naproti tomu metoda detekce pomocí anomálií vychází především z analýzy provozu. Typicky jde o stanovení mezí, které definují normální provoz na základě určitého rozložení.

Současné systémy již integrují techniky strojového učení a to jak metody založených na učení s učitelem tak na metody bez učitele. Nejvíce používanými v případě IDS jsou algoritmy podpůrných vektorů (SVM), shluková analýza a nerunové sítě[12].

[3] Na které je například uživatel již přihlášen.
[4] Zápis obrázku v jazyce HTML
[5] Server B je zde oběť útoku.

elektrorevue
ISSN 1213 - 1539

2.2.1 Gaussovo rozdělení

K základnímu popisu normálního chování je vhodné použít spojitou náhodnou veličinu. Dobrým příkladem tohoto modelu je Gaussovo rozdělení. Právě normální rozdělení pravděpodobnosti dobře aproximuje síťový provoz z hlediska výskytu anomálií. Vztah je vyjádřen funkcí 1.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \tag{1}$$

kde μ představuje střední hodnotu normálního rozdělení viz 2 a σ reprezentuje směrodatnou odchylku [15].

$$E(x) = \mu. \tag{2}$$

Rozptyl je poté dán umocněním směrodatné odchylky 3.

$$D(x) = \sigma^2. \tag{3}$$

2.2.2 Metriky binární klasifikace

Pro měření přesnosti rozpoznávání anomálií se využívá jednoduchá binární klasifikace. Byly zavedený konkrétní příznaky pro popis provozu, které indikují správný či nesprávný záchyt apod.

Veškeré míry uváděné níže využívají následující značení: TP (Skutečně pozitivní – *True positive*), FP (Falešně pozitivní – *False positive*), FN (Falešně negativní – *False negative*), TN (Skutečně negativní – *True negative*).

TPR (Míra skutečné pozitivity – *True Positive Rate*) – se vypočítává jako poměr mezi počtem správně předpovězených útoků a celkovým počtem útoků. Pokud jsou detekovány všechny narušení, pak je TPR 1, což je například pro IDS extrémně vzácné. TPR se také nazývá Detection Rate (DR) nebo citlivost. TPR lze matematicky vyjádřit vztahem 4[11].

$$TPR = \frac{TP}{TP + FN}. \tag{4}$$

FPR (Míra falešné pozitivity – *False Positive Rate*) – vypočítává se jako poměr mezi počtem normálních instancí nesprávně klasifikovaných jako útok a celkovým počtem normálních instancí viz 5.

$$FPR = \frac{FP}{FP + TN}. \tag{5}$$

FNR (Míra falešné negativity – *False Negative Rate*) – jedná se o situaci, kdy detektor nedokáže určit anomálii a klasifikuje ji jako normální stav. FNR lze matematicky vyjádřit vztahem 6.

$$FNR = \frac{FN}{FN + TP}. \tag{6}$$

CR (Přesnost – *Classification rate*) – měří, jak přesný je konkrétní systém při detekci normálního nebo abnormálního chování provozu. Je popsán jako procento všech těchto správně předpovězených instancí na všechny instance. Je definován vztahem 7.

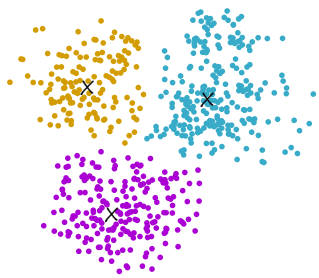
$$CR = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

2.3 Strojové učení

2.3.1 Shluková analýza

Jedná se o statistickou klasifikační metodu, která při zpracování obrazu sdružuje pixely podobných vlastností do shluků či tzv. klastrů. Každý shluk má poté nějaké společné vlastnosti, které jsou vždy odlišné od vlastností u jiných shluků. Podobnost data v rámci klastru je maximální a naproti tomu odlišné shluky zase minimální, dochází zde tedy k vytvoření smysluplných struktur na základe vlastností daného klastru [10]. Shlukování je použito například u neuronových sítí v případě učení bez učitele [7].

Jednou z možností, jak data shlukovat, je metoda *k-means* demonstrovaná na obrázku 1. Jedná se o iterativní algoritmus, kde počet shluků `k` je předem dán. Jsou zavedeny centrální body resp. středy shluků, které reprezentují shluk a jsou také ve stejném prostoru jako vstupní množina dat. Centrální body jsou obvykle voleny náhodně nebo s využitím konkrétní heuristiky. Kolem zvoleného centrálního bodu se ostatní body zařazují dle nejmenší vzdálenosti[6]. Jakmile dojde ke klasifikaci dle stanoveného kritéria, tak je poté znova přepočítán i středový bod.



Obrázek 1: K-means.

Vzdálenosti dvou bodu musí splňovat podmínky symetrie, identity, nezápornosti a trojúhelníkové nerovnosti.[10]. Jednou ze základních metrik výpočtu vzdálenosti dvojice bodů je Euklidovská metrika dle vztahu 8.

2.3.2 K-NN

K-NN (K-nejbližších sousedů – *K-Nearest Neighbors*) – jedná se algoritmus učení s učitelem [14]. K-NN klasifikuje data na základě dat sobě si nejvíce podobných. Jedná se neparametrickou metodu klasifikace, která je velice závislá na kvalitě datasetu. Metoda má problémy v rozhodování především v hraničních oblastech. [16]. Orientační chyba K-NN je za určitých podmínek srovnatelná s neuronovými sítěmi. K-NN není schopná generalizace [5].

[6] Využívá se více metrik výpočtu vzdálenosti dvojice bodů.

elektrorevue
ISSN 1213 - 1539

K-NN musí reinterpretovat data tak, aby bylo možné pracovat se vzdáleností mezi samotnými body. Nejběžnějším způsobem je Euklidovská vzdálenost, která bude definována vztahem 8.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}, \tag{8}$$

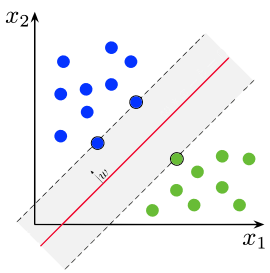
kde funkce $d()$ symbolizuje vzdálenost a p a q značí body v Euklidovském prostoru. Po výpočtům vzdálenosti mezi body dojde k výpočtu pravděpodobnosti, do jaké skupiny by měl daný bod být zařazen.

2.3.3 SVM

SVM (Algoritmy podpúrných vektorú – *Support vector machines*) – jedná se o typ algoritmu s učitelem, který je využíván především pro řešení klasifikačních čí regresních úloh. Primárně je ale využíván ve strojovém učení pro klasifikaci. Hlavním cílem SVM je oddělit data pomocí lineární hranice a využitím multidimenzionálního prostoru. Typický je třeba tento vícedimenzionální prostor vytvořit a díky tomu je již možné data roztřídit do tříd. Díky hranici, které je tvořena podpůrnými vektory, je pak možné data již zařazovat do správné kategorie. Podpůrné vektory zde mají roli extrémů, které vytyčují hranice.

$$ax + by + c = 0. \tag{9}$$

Klasifikátor lze matematicky popsat vztahem 9 [17]. Ukázka klasifikace do dvojrozměrného prostoru je demonstrována na obrázku 2.



Obrázek 2: SVM.

3 Simulace útoku a návrh řešení

Následující část se zaměřuje na aktuální hrozby, které byly objeveny v PHP aplikacích. Princip útoku bude podrobně popsán, otestován a nasimulován jeho možný dopad na reálných aplikacích či webových serverech.

3.1 Popis simulace

3.1.1 Testovaná hrozba

Konkrétní zranitelnost nese označení CVE-2020-15227[7]. Jedná se chybu v PHP frameworku Nette, který je ve velké míře používány v České republice a na Slovensku. Nette pohání velké české a slovenské projekty. Zranitelnost je typu injekce resp. A1 ze seznamu OWASP. Jedná se o typ kódové injekce a zneužití této zranitelnost může vést až k vykonání vzdáleného kódu. Tato zranitelnost byla objevena koncem roku 2020 a do současnosti se stále evidují pokusy o zneužití této chyby. Ovlivněny byly verze před nasazením následujících verzí frameworku: 2.0.19, 2.1.13, 2.2.10, 2.3.14, 2.4.16, 3.0.6.[3].

V následující části budou vytvořeny případy, které dokáží danou chybu využít a tím tak demonstrovat, co vše by daná chyba mohla způsobit. Simulace bude implementovat i jeden z nejhorších možných případů, tedy destruktivní útok, který bude mít za následek nevratné smazání aplikace pomocí vykonání vzdáleného kódu.

3.1.2 Postup simulace útoku

Útok je realizován pomocí speciálně sestavené URL, která se následně ve více modifikacích rozdistribuuje na každý kontejner webové aplikace. Jelikož každý kontejner bude obsahovat jinou verzi, může dojít k odlišné reakci na útok. V prvním kroku bude nutné sestavit již avizované prostředí, jeho architektura je blíže popsána v následující kapitole 3.1.3. Dalším krokem je sestavení speciální URL, k tomuto účelu bude realizován v sekci 3.2 testovací software v jazyce python, který bude obsahovat slovník různých modifikací útoků a tento útok na popsané prostředí provede. URL bude sestavena kombinací query stringu a posloupnosti příkazů jazyka PHP, který se má v dané injekci provést. Posledním krokem bude zjištění zdali se útok a případná analýza dopadu kódové injekce.

3.1.3 Architektura prostředí

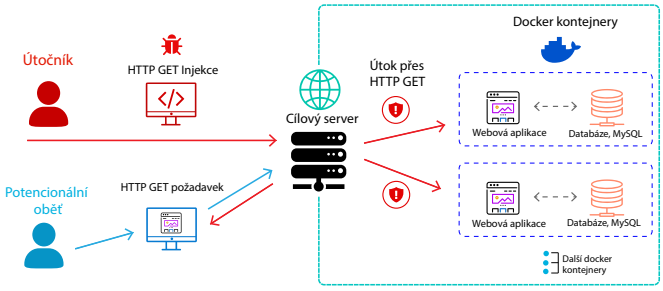
Architektura se skládá z potencionální oběti útoku a útočníka. Architektura je navržena tak, aby simulovala a reflektovala současný stav nasazování aplikací do produkce. Potencionální oběť může a nemusí být útokem ovlivněna. Útočník pomocí speciálně sestavené URL (Webová adresa – *Uniform Resource Locator*) pošle `HTTP GET` požadavek na zranitelný server. Na tomto serveru bude běžet více verzí webových aplikací postavených na Nette frameworku. Paralelní běh bude realizován více instancemi v docker kontejneru. Každý docker kontejner bude mít jinou verzi Nette, která by měla podle CVE (Běžné zranitelnosti – *Common Vulnerabilities and Exposures*) obsahovat konkrétní zranitelnost.

V kontejneru se nachází zdrojové kódy PHP aplikace. HTTP požadavky dané webové aplikace odbavuje WWW

[7] Značení se skládá z prefixu CVE, což je projekt pro identifikaci zraniltností, dále čtyřmístné číslice roku ve formátu YYYY a dále sekvenční číslo dané zranitelnosti.

elektrorevue
ISSN 1213 - 1539

server Apache na lokální úrovní. Pro ukládání perzistentních dat obsahuje kontejner MySQL databází. Celá architektura je znázorněná na obrázku 3.



Obrázek 3: Konfigurace prostředí.

3.2 Realizace programu k vykonání útoku

Na základě popsaného zranitelnosti a principu útoku byl realizován automatický nástroj v jazyce Python, který zranitelnost dokáže detekovat pomocí předpřipravených testů. Jednotlivé testy jsou uloženy ve slovníku, kde jsou rovněž další modifikace konkrétní zranitelnosti.

3.2.1 Testované útoky

Všechny testované útoky jsou obdobou vzdáleného vytvoření `.php` souboru. Do vytvořeného souboru je pak vložen jednoduchý testovací PHP skript s razítkem. V těchto případech je provedeno vzdálené vytvoření souboru a vypsání hodnoty. Jeden z útoků implementuje již problematičtější injekci. Jak lze pozorovat z výpisu 1. Jedná se o funkci, která se pokusí rekurzivně smazat konkrétní složku.

```
<?php
function removeDir($dirPath){
  if (is_dir($dirPath)){
    $objects = scandir($dirPath);
    foreach ($objects as $object){
      if ($object != '.' && $object !='..'){
        if (filetype($dirPath.'/'.$object) != 'dir'){
          unlink($dirPath.'/'.$object);
        }
        else{
          removeDir($dirPath.'/'.$object);
        }
      }
    }
    reset($objects);
    rmdir($dirPath);
  }
}
?>
```

Výpis 1: PHP vložený kód.

Nette aplikace obsahují v kořenovém adresáři mj. složku `/www`, která je přístupná veřejně. Všechny požadavky z webové serveru by měly přes konfigurační soubor `.htaccess` směřovat právě do této složky. Ve složce `/www` dochází jak k zavolání celého frameworku s danou aplikací, tak dále jsou zde obsaženy veřejně přístupná data jako jsou obrázky, javascriptové soubory, kaskádové styly apod. Tento PHP script se této informace pokusí využít k tomu, aby tuto složku rekurzivně smazal včetně veškerého obsahu. V případě úspěšného provedené dojde k nevratnému poškození aplikace a jejímu následnému pádu.

Před vložením PHP funkce do hodnoty query parametru je nutné ze scriptu odstranit bíle znaky a upravit dle požadavků kladených na URL viz níže.

3.3 Vykonání útoku a jeho dopad

3.3.1 HTTP GET požadavek

Aby bylo možné poslat `HTTP GET` požadavek, je nutné sestavit řetězec dotazu(*query string*). V případě GET postačuje k provedení útoku pouze adresa URL. Výsledná URL se skládá z koncového bodu Nette aplikace a query stringu. Mimo koncový bod sítě a nastavení klíčů bude řetězec dotazu obsahovat zdrojový kód PHP, který bude upraven do speciální podoby tak, aby injekce mohla proběhnout. URL je nutné před přiřazení hodnoty danému klíči v query stringu ještě upravit dle podmínek stanovenými RFC (Dokumenty pro popis internetových protokolů – *Request For Comments*). RFC tyto podmínky nenařizuje, ale implementace protokolů se jimi řídí. Některé symboly adres, které zároveň vyžadují programovací jazyky ve své syntaxi, jsou obecně v URL považovány za nespolehlivé a je tedy nutný jejich převod do ASCII (Americký standardní kód pro výměnu informací – *American Standard Code for Information Interchange*). Nespolehlivé znaky jsou převedeny dle tabulky do formátu $\%xx$ kde xx je hexadecimální číslice zástupného znaku. Bez této konverze nebude útok úspěšný. [8]. Sestavený požadavek obsahující útok je demonstrován ve výpisu 2[9].

```
GET /nette.m[...] HTTP/1.1
Host: localhost
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Authorization: Basic cGVudGFkbWluOnBlbnRhZG1pbg==
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,[...]
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en,cs-CZ;q=0.9,cs;q=0.8,fr;q=0.7 [...]
Cookie: nette-samesite=1; SID=6bWB4Jp474[...]
```

Výpis 2: Požadavek typu GET obsahující zranitelnost.

3.3.2 HTTP GET odpověď

Za normálních okolností jsou příchozí požadavky s kódem **2XX** považovány za úspěch, bohužel v případě útoku to již není dobrý signál. Odpověď lze vidět ve výpisu 3.

Pomocí testovacího programu byly provedeny útoky na testované verze Nette ve verzích v3.0.4, v2.4.0 a v2.0.0.

[8] Součást URL.

[9] Požadavek byl zjednodušen do následující podoby nahrazením delších sekvencí pomocí [...]. Nahrazení zástupnými symboly je použito převážně z důvodu bezpečnosti.

elektrorevue
ISSN 1213 - 1539

V případě GET požadavku obsahujícím útok došlo k navrácení HTTP status kódu 200. V případě POST požadavku byl navrácen kód 500 a bylo zjištěno, že daná zranitelnost žádný systém neovlivnila. V případě již zmíněného GET došlo k vytvoření .php souboru ve vzdálené aplikaci a k následnému spuštění. Třetí test v pořadí již obsahoval destrukční PHP script 1 a po zavolání vytvořeného souboru PHP došlo k rekurzivnímu smazaní celé frontendové prezentace. Následkem tohoto útoku tedy došlo k pádu celé aplikace. Malou úpravou by pravděpodobně šlo útok modifikovat k rekurzivnímu smazán i celé aplikace případně doimplementovat databázový příkaz ke smazání všech produkčních dat apod.

```
HTTP/1.1 200 OK
Date: Thu, 29 Apr 2021 12:46:38 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.2.30
X-Powered-By: Nette
X-Frame-Options: SAMEORIGIN
Set-Cookie: nette-samesite=1; path=/; SameSite=Strict;
Set-Cookie: SID=6bWBVTW07e3Cn[...]
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: Max-Age=0; path=/; HttpOnly
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

Výpis 3: Odpověď na požadavek typu GET, který obsahoval zranitelnost.

Z tabulky 1 je patrné, že v případě požadavku `HTTP POST` k narušení aplikace nedošlo. Naopak v situaci, kdy k narušení došlo, byl návratový kód 200, v opačném případě aplikace vracela status 500.

#	v3.0.4	v2.4.0	v2.0.0
HTTP GET	ano	ano	ano
HTTP POST	ne	ne	ne

Tabulka 1: Výsledek simulace útoků.

4 Návrh detekčního mechanismu

Návrh se zabývá oblastí předzpracování datasetu neuronové sítě. Jsou nastíněny principy interpretace dat a jejich korelace s výsledky detekce.

4.1 Prostředí implementace

Aby bylo možné pracovat s modelem, je nutné mít připravené běhové prostředí. Jedná se o knihovny, díky kterým se dají vstupní data zpracovávat a poté s nimi dále pracovat.

4.1.1 Tensorflow

Jedná se o software s otevřeným zdrojovým kódem, který je určen pro vysoce náročné numerické operace. Díky své flexibilní architektuře umožňuje výpočty napříč různými platformami od obyčejných procesorů přes výkonné grafické karty až po speciální hardwarové výpočetní jednotky specializované na neuronové sítě. Díky API (Rozhraní pro programování aplikací – *Application Programming Interface*) lze provádět výpočty nejen na běžných počítačích, ale i clusterech, serverech a mobilních telefonech. API také poskytuje základní stavební bloky pro vytvoření trénovacích modelů. Tensorflow byl původně vyvíjen výzkumným týmem a inženýry z firmy Google, která se zabývala strojovým učením a neuronovými sítěmi [1]. Software je užitečný ale i v dalších vědních oblastech.

4.1.2 Keras

Knihovna Keras je stejně jako Tensorflow software s otevřeným zdrojovým kódem. Jeho autorem je François Chollet z firmy Google. Je navržen tak, aby běžel pomocí některé z knihoven Tensorflow, Microsoft Cognitive Toolkit (CNTK) nebo Theano [8]. Keras je primárně určen a navržen tak, aby umožňoval rychlé experimentování s hlubokými neuronovými sítěmi. Velký důraz klade na to, aby byl rozšířitelný, modulární a uživatelsky přívětivý. Podporuje jak konvoluční neuronové sítě, tak rekurentní sítě. Díky vhodně navrženému API, které následuje doporučené postupy tj. široce používané návyky při jeho návrhu, je významně zredukována kognitivní zátěž [8].

4.2 Návrh datasetu

Jedním z nejdůležitějších faktorů v tomto návrhu je příprava datasetu a následná práce s ním. Celý návrh je postaven na rozdílném přístupu při předzpracování tak, aby bylo možné dosáhnout lepších výsledků. Je kladen velký důraz na kvalitu jeho předzpracování. Většina útoků, které se dějí např. pomocí metod `HTTP GET` jsou v jistých sledovaných atributech velice podobné. Bude zde velice důležité sestavit logiku pro předzpracování datasetu tak, aby vše vycházelo z podobnosti mezi útoky. Důležitý je především faktor zjevného rozeznání v případech, kdy by se jednalo o útoky, na které ještě sítě nebyly trénovány.

U tohoto typu problému je vhodné použít metodu učení s učitelem. Konkrétní předzpracování datasetu by mělo probíhat na základě pravidel nebo pokročilé gramatiky. Gramatika sama o sobě bude mít dostatečnou sílu na to, aby dokázala deterministicky popsat určité vychýlení od normálu apod.

Celý proces předzpracování bude vycházet ze znalosti url viz tabulka 2.

#	Uživatel	Heslo	Host	Port	Path	Query	Kotva
http	karel	heslo	mujweb.cz	8080	cs/home	?name=param	#anchor
https			mujweb.cz		cs/home	?name=param	#anchor

Tabulka 2: Fragmenty URL k sestavení pravidel.

Zatímco první záznam tabulky ukazuje, jak by mohla potencionálně celá adresa dle RFC v datasetu vypadat, druhý řádek symbolizuje běžný zápis, kde jméno uživatele a heslo obvykle do adres neuvádíme. Port je doplněn automaticky v případě defaultního chování. Každá část URL může být tedy teoreticky zneužita, proto by fáze předzpracování měla s tímto faktem počítat. Návrh také předpo-

elektrorevue
ISSN 1213 - 1539

kládá dataset `HTTP GET` požadavků. Soubor aplikovaných principů a pravidel půjde s drobnými úpravami aplikovat i na dataset v případě `HTTP POST` požadavků.

4.2.1 Postup předzpracování

Cílem předzpracování tedy bude interpretace konkrétní adresy a její následný převod do číselné podoby [2]. Kde konkrétní čísla by představovala výstup skeneru, tedy tokeny. Interpretace by mohla probíhat např. způsobem dle tabulky 3 níže.

Token	Hodnota	Příklad syntaxe
T_Operator	0	&&, !=
T_LanguageKeyword	1	while, function, def
T_LanguageCommand	2	unlink, .unshift()
T_Expression	3	15 & 9
T_OpenCloseTag	4	<?php, ?>, <script>
T_URLQuery	5	?q=test&id=1&p=35
T_Text	6	"Lorem ipsum ..."
T_Domain	7	example.com
...	...	...

Tabulka 3: Tokenizace datasetu.

Z tokenů bude sestaveno pole, které bude tvořit záznam v matici, která již bude představovat předzpracovaný dataset. Jinými slovy, každý `HTTP GET` požadavek bude interpretován sadou tokenů, které budou normalizovány a takto předkládány modelu neuronové sítě. Vytvoření pravidel z tokenů je zde nahrazeno neuronovou síťí.

4.2.2 Dopad na predikci

Díky takovému předzpracování datasetu postavenému na tokenové interpretaci bude možné lépe identifikovat princip hrozeb. Dataset již ve vstupním poli bude mít informaci, zdali se v URL nachází např. nebezpečné sekvence, řídící konstrukce, injekce apod. Systém intepretace pravidel do jisté míry může nahrazovat konvoluci, která dosahuje skvělých výsledků v konvolučních neuronových sítí v případech segmentace apod. Je zřejmé, že určité potencionálně nebezpečné sekvence by se mohly vyskytovat v každém běžném textu, z toho důvodu by metody fungující pouze na základě porovnávání řetězců nemohly fungovat dobře. Hlásily by příliš mnoho falešně pozitivních výsledků. Strojové učení zde může takto docílit lepšího pochopení HTTP požadavku, čímž dojde i k výrazně vyšší míře přesnosti.

5 Závěr

Tento článek se zabývá bezpečností webových aplikací v PHP. V úvodu je nastíněna aktuální problematika bezpečnosti webových aplikací. Důvody velkého rozmachu využívání takových webových systémů s tím i rostoucí rizika.

Následující část se věnuje současným řešením bezpečnostních problémů webových aplikací a dále popisuje základní principy, jak tyto systémy s hrozbami pracují. Jsou popsány některé důležité principy fungování zvolených útoků, které jsou stále aktuální a je jim tedy nutné stále přizpůsobovat vývoj webových aplikací. Podrobněji je také popsána detekce u systémů IPS/IDS a principy metod strojového učení, které se při odhalování útoků a eliminaci hrozeb v současnosti využívají.

Hlavní náplň článku se zabývá realizací nového typu útoku poslední doby. Podrobně zkoumá a testuje jeho dopad na cílový systém a dále se snaží útok generalizovat tak, aby bylo možné i jemu obdobným předcházet v budoucnu. K ověření zranitelnosti je připraveno simulační prostředí pomoci kontejnerizace. Každý kontejner obsahuje jinou verzi webové aplikace a obsahuje všechny potřebné závislosti k běhu webového prostředí. Poté je navržen program v jazyce Python, jehož cílem bylo otestovat reálný dopad této zranitelnosti. Výsledek simulace demonstruje fatální dopad tohoto útoku a v navazující části je poté navrženo řešení pomocí strojového učení. Je vybrán postup interpretace webového požadavku na posloupnost tokenových pravidel, která jsou normalizovaná. Na základě této interpretace je sestavena trénovací množina pro model neuronové sítě. Návrh tohoto řešení vycházel především ze znalosti současných systémů a jejich úskalí při detekci. Tato metoda by mohla výrazně pomoci při detekci hrozeb, které nejsou doposud známé a mohla by vést k nižší hladině falešně negativních či falešně pozitivních hlášení.

Literatura

[1] ABADI, Martin, Ashish AGARWAL, Paul BARHAM, Eugene BREVDO, Zhifeng CHEN a Craig CITRO. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [online]. tensorflow.org: Google, 2015 [cit. 2021-5-25]. Dostupné z: https://www.tensorflow.org/

[2] ABAIMOV, Stanislav a Giuseppe BIANCHI. CODDLE: Code-Injection Detection With Deep Learning. IEEE Access [online]. 2019, 7, 128617-128627 [cit. 2021-6-25]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2019.2939870

[3] CVE-2020-15227 [online]. 2020 [cit. 2021-6-25]. Dostupné z: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-15227

[4] DILSHAN JAYAWICKRAMA, Thamindu. Introduction to Anomaly Detection [online]. 2020 [cit. 2021-5-25]. Dostupné z: https://towardsdatascience.com/introduction-to-anomaly-detection-c651f38ccc32

[5] HOUDEK, Michal, Tomáš SVOBODA a Tomáš PROCHÁZKA. [online]. In: . 2001, s. 1-12 [cit. 2021-4-15]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_01/4/rpz4.pdf

[6] HODO, Elike, Xavier J. A. BELLEKENS, Andrew W. HAMILTON, Christos TACHTATZIS a Robert C. ATKINSON. Shallow and Deep Networks Intrusion De-

elektrorevue
ISSN 1213 - 1539

tection System: A Taxonomy and Survey. CoRR [online]. Abs/1701.02145. 2017 [cit. 2021-4-22]. Dostupné z: http://arxiv.org/abs/1701.02145

[7] HONZÍK, Petr. SHLUKOVá ANALýZA [online]. 2014 [cit. 2021-6-25]. Dostupné z: http://vision.uamt.feec.vutbr.cz/STU/lectures/09_Shlukova_analyza.pdf. Přednáška. Ústav automatizace a měřicí techniky, FEKT, VUT.

[8] CHOLLET, François. Deep learning v jazyku Python: knihovny Keras, Tensorflow. Přeložil Rudolf PECINOVSKÝ. Praha: Grada Publishing, 2019. Knihovna programátora (Grada). ISBN 978-80-247-3100-1.

[9] JAMAL AL-MANSOR, Mohammed a Kok BENG GAN. Intrusion Detection Systems: Principles And Perspectives. ResearchGate [online]. 2018 [cit. 2021-6-25]. Dostupné z: https://www.researchgate.net/publication/332524050_Intrusion_Detection_Systems_Principles_And_Perspectives

[10] KUČERA, Jiří. Metody kategorizace dat [online]. 2008 [cit. 2021-4-25]. Dostupné z: https://is.muni.cz/th/w8lgz/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky.

[11] KHRAISAT, Ansam, Iqbal GONDAL, Peter VAMPLEW a Joarder KAMRUZZAMAN. Survey of intrusion detection systems: techniques, datasets and challenges. [online]. 2019, , 1-22 [cit. 2021-4-25]. Dostupné z: https://doi.org/10.1186/s42400-019-0038-7

[12] LABONNE, Maxime. Anomaly-based network intrusion detection using machine learning [online]. Paris, France, 2020 [cit. 2021-3-11]. Dostupné z: https://tel.archives-ouvertes.fr/tel-02988296. Disertace. Institut Polytechnique de Paris.

[13] MARTINÁSEK, Zdeněk. Problematika logování, systémy IDS a IPS [online]. Vysoké učení technické v Brně, 2020 [cit. 2021-6-25]. Přednáška. Vysoké učení technické v Brně.

[14] NASEER, Sheraz, Yasir SALEEM, Shehzad KHALID, Muhammad Khawar BASHIR, Jihun HAN, Muhammad Munwar IQBAL a Kijun HAN. Enhanced Network Anomaly Detection Based on Deep Neural Networks. IEEE Access [online]. 2018, 6, 48231-48246 [cit. 2021-6-25]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2018.2863036

[15] Normální rozdělení pravděpodobnosti [online]. portal.matematickabiologie.cz: Institut biostatistiky a analýz LF a PřF MU, 2020 [cit. 2021-6-25]. Dostupné z: http://portal.matematickabiologie.cz

[16] SCHOTT, Madison. K-Nearest Neighbors (KNN) Algorithm for Machine Learning [online]. 2019 [cit. 2021-5-15]. Dostupné z: https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26

[17] TAHER, Kazi Abu, Billal MOHAMMED YASIN JISAN a Md. Mahbubur RAHMAN. Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection. In: 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST) [online]. IEEE, 2019, 2019, s. 643-646 [cit. 2021-6-25]. ISBN 978-1-5386-8012-4. Dostupné z: doi:10.1109/ICREST.2019.8644161

[18] Usage statistics of server-side programming languages for websites. W3Techs [online]. W3Techs.com, 2021 [cit. 2021-4-12]. Dostupné z: https://w3techs.com/technologies/overview/programming_language