2. Środowiska wirtualne

Środowisko wirtualne (ang. **Virtual Environments**, VE) to oprogramowanie symulujące działanie sprzętu komputerowego. W odróżnieniu od tradycyjnych systemów fizycznych umożliwia ono jednoczesne funkcjonowanie wielu niezależnych systemów operacyjnych lub aplikacji na jednej maszynie [1]. Działają one niezależnie od fizycznej struktury sprzętu. Tworzenie środowisk wirtualnych możliwe jest dzięki wykorzystaniu technologii wirtualizacji. Jest to technologia, wykorzystująca środowisko logiczne do przekroczenia fizycznych ograniczeń sprzętowych [2].

Podstawowym elementem środowisk wirtualnych jest maszyna wirtualna (ang. **Virtual Machine**, VM). Jest to aplikacja, wykonujące program tak, jakby była fizycznym urządzeniem, więc można byłoby powiedzieć, że jest to „komputer”, działający wewnątrz fizycznego komputera. Aplikacja VM (nazywana „gościem”) uruchamia swój własny system operacyjny na rzeczywistej maszynie (zwanej „gospodarzem”). Wirtualny system operacyjny może być dowolny, np. Windows lub MacOS, i nie jest ograniczony do jednego systemu operacyjnego na maszynie gospodarza [3]. Każda maszyna wirtualna działa niezależnie i nie ma wpływu na działanie innych VM-ów.

2.1. Charakterystyka wirtualizacji

Wirtualizacja to technologia umożliwiająca tworzenie wielu odizolowanych środowisk komputerowych – zwanych maszynami wirtualnymi (VM) – na jednym fizycznym urządzeniu. Dzięki warstwie pośredniczącej, zwanej hipernadzorcą (**hypervisor**), każda maszyna wirtualna może działać jak odrębny komputer z własnym systemem operacyjnym i aplikacjami, niezależnie od innych instancji. To podejście pozwala na efektywne wykorzystanie zasobów sprzętowych, zwiększenie skalowalności, uproszczenie zarządzania oraz ograniczenie kosztów operacyjnych [4].

Hipernadzorca (hypervisor) to oprogramowanie, które umożliwia tworzenie i zarządzanie maszynami wirtualnymi poprzez oddzielenie systemów operacyjnych gości od fizycznej infrastruktury sprzętowej. W zależności od sposobu działania, wyróżniamy dwa główne typy:

* typ 1: natywny (bare-metal) – działa bezpośrednio na sprzęcie, bez potrzeby instalowania systemu operacyjnego gospodarza. Przykładowo: Microsoft Hyper-V,
* typ 2: hostowany – funkcjonuje jako aplikacja zainstalowana w ramach istniejącego systemu operacyjnego. Przykładowo: VirtualBox, VMware Workstation.

Dodatkowo, hypervisory można klasyfikować ze względu na sposób wirtualizacji:

* Pełna wirtualizacja – system gościa działa bez konieczności modyfikacji, nieświadomy, że funkcjonuje w środowisku wirtualnym, jest w pełni niezależny.
* Wirtualizacja wspierana sprzętowo – wykorzystuje specjalne funkcje procesora, takie jak Intel VT-x czy AMD-V, w celu optymalizacji pracy maszyn wirtualnych.
* Parawirtualizacja – wymaga modyfikacji systemu gościa, który jest świadomy, że działa w środowisku wirtualnym i potrafi efektywnie współpracować z hipernadzorcą [5].

2.2. VMware jako platforma testowa

Vmware Workstation Pro (od firmy Broadcom) jest w pełni zwirtualizowanym środowiskiem sprzętowym dla systemu operacyjnego gościa. Program ten obsługuje wiele systemów operacyjnych gospodarza, w tym Windows, Linux oraz macOS, i został zaprojektowany z myślą o maksymalnym wykorzystaniu fizycznych zasobów komputera, co przekłada się na wysoką wydajność działania maszyn wirtualnych [4].

Oprogramowanie to stara się wykonywać instrukcje bezpośrednio na procesorze gospodarza, o ile jest to możliwe. W przypadkach, gdy bezpośrednie wykonanie kodu nie jest wspierane przez sprzęt, oprogramowanie korzysta z techniki dynamicznego tłumaczenia binarnego, która pozwala na przekształcanie instrukcji w czasie rzeczywistym. Tak przetworzony kod jest przechowywany w pamięci RAM, co umożliwia dalsze działanie maszyny wirtualnej z dużą szybkością – według producenta, z wydajnością sięgającą ponad 80% względem natywnego systemu [4].

VMware emuluje podstawowe elementy sprzętowe, takie jak karta graficzna (czego nie oferuje np., VirtualBox od firmy Oracle), karta sieciowa czy kontrolery dysków, a także umożliwia dostęp do urządzeń USB, portów szeregowych i równoległych za pomocą sterowników pośredniczących. Warto jednak zauważyć, że przenoszenie maszyn wirtualnych między różnymi komputerami, szczególnie z odmiennymi architekturami procesora lub liczbą rdzeni, może wymagać dodatkowej konfiguracji ze względu na różnice w zestawach instrukcji [4].

Dzięki powyższym rozwiązaniom, VMware Workstation Pro wyróżnia się wysoką wydajnością i wszechstronnością, co czyni go solidnym narzędziem do prowadzenia badań nad bezpieczeństwem w środowiskach wirtualnych.

A screenshot of a computer

AI-generated content may be incorrect.

Fot. 1 Widok podstawowy Vmware Workstation w wersji 17.6.3

Podstawowymi funkcjami dostarczanymi przez środowisko Vmware Workstation są m.in.:

- tworzenie maszyn wirtualnych z wybraną konfiguracją systemu operacyjnego, pamięci RAM, przestrzeni dyskowej,

- snapshoty, czyli zapisywanie stanu maszyny wirtualnej w wybranym przez użytkownika momencie, co daje możliwość przywrócenia zapisanego obrazu,

- klonowanie maszyn wirtualnych,

- uruchamianie maszyn wirtualnych jednocześnie przy jednoczesnym ich izolowaniu.

- zaawansowana konfiguracja sieciowa, pozwalająca na definiowanie topologii sieci wirtualnych (m.in. NAT, bridge, host-only)

- połączenia ze zdalnymi serwerami, np. z Vmware ESXi.

2.3 Konfiguracja środowiska testowego

Do przeprowadzenia badań wykorzystano komputer stacjonarny wyposażony w procesor AMD Ryzen 5 3600 (6 rdzeni, 12 wątków), 64 GB pamięci RAM DDR4 (3600 MHz), dysk SSD M.2 o pojemności 2 TB oraz kartę graficzną NVIDIA GeForce RTX 3060 Ti. Systemem operacyjnym gospodarza był Windows 11 Pro (wersja 24H2).

Środowisko wirtualne zostało zrealizowane przy użyciu VMware Workstation Pro 17.6.3. W ramach testów utworzono dwie maszyny wirtualne: jedną z systemem Windows 10 Pro (wersja 22H2, 64-bit), drugą z systemem Ubuntu 24.04.2 LTS. Obie maszyny zostały skonfigurowane zgodnie z poniższymi założeniami:

* **Procesor**: przypisano 4 rdzenie wirtualne w celu zapewnienia stabilności i wydajności przy jednoczesnym zachowaniu zasobów hosta.
* **Pamięć RAM**: każdej maszynie przydzielono 8 GB pamięci operacyjnej.
* **Typ połączenia sieciowego**: zastosowano tryb mostkowany (bridged), umożliwiający pełną komunikację z innymi urządzeniami w sieci lokalnej, co było niezbędne dla testów typu Man-in-the-Middle czy sniffing.
* **Typ dysku**: dla systemu Windows wybrano nośnik NVMe, natomiast dla Ubuntu – SCSI, zgodnie z rekomendacjami instalatorów i wymaganiami kompatybilności.
* **Typ oprogramowania układowego**: wybrano UEFI ze względu na nowoczesne mechanizmy bezpieczeństwa oraz zgodność z funkcjami systemowymi, takimi jak Microsoft Defender Credential Guard.
* **VMware Tools**: zainstalowano na obu maszynach w celu poprawy wydajności i zapewnienia pełnej integracji ze środowiskiem wirtualnym.

Maszyny wirtualne zostały przygotowane w sposób umożliwiający swobodną realizację zaplanowanych scenariuszy testowych. Ich konfiguracja została dostosowana do wymagań badań nad bezpieczeństwem systemów operacyjnych, z uwzględnieniem typowych ustawień spotykanych w warunkach rzeczywistych.

2.4 Kali Linux – narzędzie do przeprowadzania testów

Na osobnym komputerze podłączonym do tej samej sieci zainstalowany został system Kali Linux w wersji 2025.1c. Jest to dystrybucja typu open-source oparta na systemie Debian, stworzona przez Offensive Security w celach przeprowadzania testów penetracyjnych oraz audytów bezpieczeństwa. Posiada on wbudowane narzędzia dedykowane testom bezpieczeństwa, a także dostosowane jądro systemu do wstrzykiwania różnego rodzaju pakietów [8].

System ten został wybrany do testów, ze względu na swoją popularność, stabilność oraz szeroką społeczność skupioną wokół oprogramowania. Jest to adekwatne narzędzie do przeprowadzenia testów bezpieczeństwa w środowiskach wirtualnych.

4. Testowanie bezpieczeństwa - ataki sieciowe

4.1 Sniffing

Celem testów było sprawdzenie czy możliwe jest pasywne podsłuchiwanie ruchu sieciowego (sniffing) z systemu Kali Linux zainstalowanego na fizycznym komputerze, przy użyciu narzędzia Wireshark. Wszystkie urządzenia były połączone do tej samej sieci lokalnej (Wi-Fi). Maszyny testowe – zarówno fizyczne, jak i wirtualne – pracowały w trybie mostkowanym (bridged), co zapewniało im bezpośrednią obecność w tej samej podsieci IP co komputer atakujący.

W ramach badania przeprowadzono cztery testy: dwa z systemem Windows 10 (na maszynie fizycznej i wirtualnej) oraz dwa z systemem Ubuntu 22.04 (również fizycznie i w VM). We wszystkich przypadkach wyłączono domyślne zabezpieczenia systemowe, takie jak zapora Windows Firewall czy ufw, aby sprawdzić, czy ruch HTTP da się przechwycić w warunkach minimalnej ochrony.

Tabela poniżej przedstawia zestawienie przeprowadzonych testów:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | System atakowany | Środowisko | Zabezpieczenia | Ruch HTTP przechwycony |
| WV | Windows 10 | VMware | brak | nie |
| WB | Windows 10 | Bare-metal | brak | nie |
| UV | Ubuntu 22.04 | VMware | brak | nie |
| UB | Ubunu 22.04 | Bare-metal | brak | nie |

Tab. X. Testy przeprowadzone w ramach ataku typu sniffing

Program Wireshark w wersji 4.4.4 został uruchomiony na interfejsie sieciowym wlan0[[1]](#footnote-1). Włączony został tryb promiscous.[[2]](#footnote-2) W celu zawężenia analizy, w Wiresharku ustawiono filtr ograniczający rejestrowane pakiety wyłącznie do protokołu HTTP. Test polegał na odwiedzeniu strony HTTP (http://testphp.vulnweb.com/login.php)[[3]](#footnote-3) na maszynie ofiary oraz zasymulowaniu ruchu poprzez wprowadzenie i wysłanie przykładowych danych logowania.

A screenshot of a computer

AI-generated content may be incorrect.  
Fot. Konfiguracja filtrowania ruchu HTTP w narzędziu Wireshark na interfejsie wlan0.

Po uruchomieniu przechwytywania i wygenerowaniu ruchu z maszyn ofiar Wireshark nie zarejestrował żadnego ruchu HTTP w żadnym z testowanych przypadków.

A screenshot of a computer

AI-generated content may be incorrect.  
Fot. X. Brak przechwyconego ruchu HTTP po stronie atakującego w przypadku ruchu z maszyny ofiary.

Dla porównania i zobrazowania ataku, a także sprawdzenia metodologii, przeprowadzony został dodatkowy test, gdzie połączenie http odbyło się na komputerze atakującym. Wireshark odnotował ruch sieciowy.

A screenshot of a computer

AI-generated content may be incorrect.  
Fot. Przechwycony ruch HTTP po wygenerowaniu go lokalnie na komputerze atakującym.

Dane przesyłane w formularzu logowania zostały przeanalizowane przy użyciu funkcji „Follow HTTP Stream”, która umożliwiła pełny podgląd przesłanych treści, w tym danych logowania.

A screenshot of a computer

AI-generated content may be incorrect.  
Fot. Podgląd danych logowania uzyskanych przez narzędzie „Follow HTTP Stream” w Wireshark na komputerze atrakującym.

****

**Rys. x.** Schemat topologii środowiska testowego w ataku sniffingu.

Przygotowany został schemat przedstawiający topologię środowiska testowego. W jego skład wchodzą: maszyny testowe (fizyczne oraz VM), router Wi-Fi (pełniący rolę przełącznika sieciowego) oraz komputer atakujący z Kali Linux i Wiresharkiem. Ruch sieciowy z maszyn ofiar nie był kierowany do atakującego, ponieważ przełącznik przekazuje pakiety wyłącznie do docelowego adresata, eliminując możliwość pasywnego podsłuchu.

Brak przechwyconego ruchu HTTP nie świadczy o błędach w konfiguracji testu, lecz o skuteczności współczesnej architektury sieci w ograniczaniu dostępu do pakietów przez nieuprawnione hosty. W kontekście bezpieczeństwa oznacza to, że nawet przy wyłączonych zaporach systemowych, izolacja sieciowa skutecznie chroni przed pasywnym sniffingiem. W środowiskach wykorzystujących przełączniki sieciowe (np. domowe routery Wi-Fi) standardowy sniffing z użyciem narzędzi takich jak Wireshark nie pozwala na podsłuchiwanie ruchu pomiędzy innymi urządzeniami w sieci. Wynika to z faktu, że przełączniki przekazują pakiety tylko do właściwego odbiorcy, co uniemożliwia pasywne przechwytywanie pakietów przez inne hosty [9].

Z przeprowadzonych badań wynika, że w przypadku testów sniffingu maszyny wirtualne nie wykazują różnicy w kontekście bezpieczeństwa w porównaniu do maszyn fizycznych. Ustawienie urządzeń w tej samej sieci (typu bridged) nie wystarczyło do przechwycenia ruchu http z maszyny ofiary.

Testy w wariantach z aktywnymi zabezpieczeniami nie zostały przeprowadzone – skoro przechwycenie ruchu było niemożliwe w najmniej chronionym scenariuszu, dalsze zwiększanie ochrony nie miałoby wpływu na wynik. Wnioskiem praktycznym jest to, że skuteczne przechwytywanie ruchu HTTP w takich warunkach wymaga zastosowania ataku aktywnego, np. typu ARP spoofing lub Man-in-the-Middle.

4.2 ARP spoofing

4.2.1 ARP spoofing teoretycznie

**ARP spoofing** to technika ataku w sieciach lokalnych, polegająca na wysyłaniu fałszywych komunikatów ARP (Address Resolution Protocol), które mają na celu przekonanie urządzenia ofiary, że atakujący jest innym zaufanym hostem — najczęściej bramą sieciową. Dzięki temu możliwe jest przechwycenie, zmodyfikowanie lub przekierowanie ruchu sieciowego [10].

Adresy IP służą do identyfikacji urządzeń w sieci, ale na poziomie fizycznym komunikacja odbywa się za pomocą adresów MAC. Aby powiązać adres IP z adresem MAC, urządzenia korzystają z ARP, które dynamicznie buduje tzw. tablicę ARP — mapującą IP do MAC. Atakujący może wykorzystać ten mechanizm, wysyłając spreparowaną odpowiedź ARP, która podszywa się pod router lub inne urządzenie. Celem jest wpisanie fałszywego powiązania IP ↔ MAC w tablicy ARP ofiary [10].

Atak przebiega zazwyczaj w dwóch etapach:

1. Atakujący wysyła fałszywe odpowiedzi ARP do ofiary, informując, że adres IP routera (bramy) należy do jego (atakującego) adresu MAC.
2. Ofiara aktualizuje swoją tablicę ARP i zaczyna przesyłać dane do atakującego zamiast do faktycznej bramy. Atakujący może przechwycić dane, a następnie przesłać je dalej (atak typu Man-in-the-Middle).

Ataki typu ARP spoofing są popularne nie tylko ze względu na swoją skuteczność, ale również z powodu prostoty implementacji. Wystarczy, że atakujący znajdzie się w tej samej sieci lokalnej co ofiara i będzie w stanie wysyłać pakiety ARP – nie są potrzebne żadne specjalne uprawnienia [11].

Z perspektywy bezpieczeństwa, ARP spoofing stanowi zagrożenie, ponieważ może prowadzić do:

* kradzieży tożsamości użytkownika (np. przejęcie sesji logowania),
* instalacji złośliwego oprogramowania (poprzez modyfikację ruchu),
* ataków typu DoS (gdy dane nie są przekazywane dalej po przejęciu).

Atak ARP spoofing pozwala nie tylko na pasywne podsłuchiwanie ruchu, ale także na jego modyfikowanie. Atakujący może przechwycić dane logowania, sesje HTTP, a nawet wstrzykiwać złośliwy kod [10].

4.2.2 Atak ARP spoofing

Przeprowadzone próby ataku ARP spoofing miały na celu przechwycenie danych logowania na testowanych systemach operacyjnych, z wykorzystaniem techniki **Man-in-the-Middle (MITM)**. Atak realizowany był za pomocą narzędzia **Ettercap**, a do analizy przechwyconego ruchu użyto dodatkowo **Wiresharka**.

Mechanizm ataku opierał się na fałszowaniu odpowiedzi ARP w lokalnej sieci, dzięki czemu maszyna atakująca (Kali Linux) była rozpoznawana przez system ofiary jako domyślna brama sieciowa. W wyniku tego cały ruch sieciowy ofiary był przekierowywany przez komputer atakującego. Pozwoliło to na przechwytywanie żądań HTTP, a w niektórych przypadkach – danych logowania przesyłanych w postaci nieszyfrowanej.

W trakcie każdego testu użytkownik na maszynie ofiary otwierał stronę <http://testphp.vulnweb.com/login.php> i ręcznie wpisywał dane logowania (np. admin:test). Przeprowadzano **po 15 prób dla każdego scenariusza** testowego, z rejestracją, czy dane zostały skutecznie przechwycone, czy tylko widoczny był ruch sieciowy, czy też nie odnotowano żadnej aktywności.

W przypadku Ettercapa, dane logowania (login i hasło) były często wyświetlane bezpośrednio w dolnym panelu aplikacji, w zakładce *"Messages"* lub *"Connections"* – jako fragment przesyłanego żądania HTTP POST. Narzędzie to potrafi również filtrować pakiety według treści i rozpoznawać typowe formularze logowania, dzięki czemu nawet bez szczegółowej analizy można było szybko ocenić, czy atak zakończył się sukcesem.

W Wiresharku natomiast dane logowania były identyfikowane jako treść żądania HTTP – widoczne w zakładce *"Follow HTTP Stream"*, najczęściej w formacie:

username=admin&password=test

lub jako fragment application/x-www-form-urlencoded w pakiecie typu POST. Widoczność tych danych była uzależniona od warunków testu – systemu, środowiska uruchomieniowego (VM/bare-metal) oraz aktywnych zabezpieczeń.

Scenariusze testowe obejmowały osiem konfiguracji: Windows 10 oraz Ubuntu 22.04, każdorazowo uruchomione jako maszyna wirtualna oraz jako system fizyczny (bare-metal), z włączonymi i wyłączonymi domyślnymi zaporami sieciowymi (firewall i ufw).

Za **skuteczny atak** uznawano przechwycenie danych logowania – niezależnie od tego, czy były widoczne w Ettercapie czy Wiresharku. Jako **częściowo skuteczny** odnotowywano przypadki, gdy pojawił się ruch HTTP, ale bez treści logowania. Brak ruchu lub brak przechwyconych pakietów był interpretowany jako **niepowodzenie**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| System | Środowisko | Zabezpieczenia | Powodzenie | Przechwycenie samego ruchu | Brak wyników |
| Windows 10 | fizyczny komputer | włączone | 11 | 4 | 0 |
| Windows 10 | fizyczny komputer | wyłączone | 9 | 6 | 0 |
| Windows 10 | maszyna wirtualna | włączone | 11 | 4 | 0 |
| Windows 10 | maszyna wirtualna | wyłączone | 12 | 3 | 0 |
| Ubuntu 22.04 | fizyczny komputer | włączone | 9 | 6 | 0 |
| Ubuntu 22.04 | fizyczny komputer | wyłączone | 6 | 9 | 0 |
| Ubuntu 22.04 | maszyna wirtualna | włączone | 6 | 8 | 1 |
| Ubuntu 22.04 | maszyna wirtualna | wyłączone | 9 | 6 | 0 |

|  |  |
| --- | --- |
| Win bare on | bare off |
| 1 ruch | oba |
| 2 oba | ruch |
| 3 ruch | ruch |
| 4 oba | oba |
| 5 oba | ruch |
| 6 oba | ruch |
| 7 oba | ruch |
| 8 ruch | oba |
| 9 oba | ruch |
| 10 oba | oba |
| 11 oba | oba |
| 12 oba | oba |
| 13 oba | oba |
| 14 oba | oba |
| 15 ruch | oba |

|  |  |
| --- | --- |
| Win vm on | vm off |
| oba | oba |
| oba | oba |
| ruch | ruch |
| oba | oba |
| oba | oba |
| oba | oba |
| ruch | oba |
| oba | ruch |
| oba | oba |
| oba | oba |
| oba | oba |
| oba | oba |
| oba | ruch |
| ruch | oba |
| ruch | oba |

|  |  |
| --- | --- |
| ubu vm on | vm off |
| oba | oba |
| oba | oba |
| ruch | ruch |
| ruch | eter |
| ruch | oba |
| eter | oba |
| OBA | ruch |
| ruch | oba |
| nic | ruch |
| ruch | oba |
| ruch | ruch |
| oba | ruch |
| ruch | eter |
| ruch | ruch |
| oba | eter |

Bibliografia

[1] Virtualization Throughout the Software Lifecycle, Sarah N. Crutchfield

[2] Virtualization and Security Aspects: An Overview, Rui Filipe Pereira, Rui Miguel Silva & João Pedro Orvalho

[3] Virtualization and Forensics A Digital Forensic Investigator’s Guide to Virtual Environments

[4] VMware vSphere Essentials: A Practical Approach to vSphere Deployment and Management Luciano Patrão

[5] Optimal guest file system for type-2 hypervisorbased virtualization in Virtual box

[6] <https://uefi.org/specs/PI/1.8/V2_Overview.html>

[7] UEFI Memory Forensics: A Framework for UEFI Threat Analysis Kalanit Suzan Segal∗

[8] <https://www.kali.org/docs/introduction/what-is-kali-linux/>

[9] Patel, N. P., Patel, R. G., & Patel, D. R. (2009). Packet Sniffing: Network Wiretapping. W: 2009 IEEE International Advance Computing Conference (IACC 2009) (s. 2691-2696). Patiala, Indie: IEEE.

[10] Daniel G. Graham, *Ethical Hacking*, No Starch Press, 2021, rozdział 2.

[11] Alina Alina; Shipra Saraswat, *Understanding, Implementing and Combating Sniffing and ARP Spoofing*. 4th Int. Conf. on Recent Developments in Control, Automation & Power Engineering (RDCAPE), 2021.

- podrozdział dlaczego win I linux i dlaczego w takich wersjach

- ‘wykorzystanie istniejących narzędzi, wyodrębnienie obszarów bezpieczeństwa, przeprowadzenie testów, podsumowanie i wnioski’

- Uzasadnienie czemu te a nie inne narzędzia

- Mocne wyodrębnienie obszarów bezpieczeństwa

- Kompleksowa ocena bezpieczeństwa systemów wirtualnych

- Które systemy na jaki typ ataków

Ubuntu: vmtestuser, password

Vmtest, password

student, password

Kali: kaliuser, password

Opis problemu, narzędzi, wprowadzenie ogólnikowe, co będzie robione – w jaki sposób, testy wydajnościowe, po wpr – krotki wstep teoretyczny

Najpierw narzędzia, przegląd literatury, istniejące rozwiązania, motywacje pracy,

Graficzna prezentacja platrofa testowa

20 sierpnia ALFA

Promotor uwagi:

~~- instrukcja obsługi – przerobić – wyrzucić zrzuty. Podstawowe informacje o maszynie i podzespołach – procesor, ram, grafika, dysk – na czym to było~~

- krótkie wprowadzenie

- układ: wprowadzenie (2str max – opis problemu, narzędzia istniejące, skoncentrowałem się na konkretnych elementach bezpieczeństwa tego systemu, będę testował w taki sposób; coś na styl streszczenia krótkiego) – co będzie robione, w jaki sposób, jakie będą efekty – czego się spodziewam w jakiej formie

- wstęp teoretyczny: teoria nt środowiska wirtualnego, bezpieczeństwa, jakich pojęć będę używał, opis narzędzi/języków (taki a taki system, czym się charakteryzuje)

- przegląd literatury, istniejące rozwiązania, MOTYWACJE PRACY

- teza i cel pracy (zakładamy tezę: np. bezpieczeństwo systemów operacyjnych jest bardzo wymagane i trudne do osiągnięcia, więc trzeba takie systemy zabezpieczyć; zakładam, że można za pomocą takiego a takiego narzędzia zaprezentować metody w jaki sposób zabezpieczyć swoje systemy; za pomocą testów opiszę na czym idea bezpieczeństwa polega) – max. pół strony (osobno teza i cel)

Cel: realizacja własnoręcznie wirtualnej maszyny z systemem takim i przeprowadzenie testów, żeby przedstawić w jaki sposób to bezpieczeństwo działa

Bezpieczenstwo sys oper jest trudje do osiagnieca w srod wirt. - teza

Realizacja wlasnorecznej maszyny z systemtem takim a takim – cel

- Czesc badawcza: jak najwięcej testów; też osobny rozdział stricte tylko testów na maszynie bez porównania też; dużo wyników i wniosków

- typy ataków teoretycznie – opisać! Zacząć od tego w części badawczej. Zrobić abstrakt graficzny – graficznie zaprezentować platformę testową (komputer z systemem rzeczywistym, drugi z VM – SCHEMAT (jednym z modułów było testowanie); nie trzeba się bawić w UML – wystarczy na kształtach – dużo kolorów – DUŻO SCHEMATÓW

Opisać na początku części badawczej z jakich elementów/modułów składa się platforma testowa – ogólny zarys żeby ktoś widział stanowisko oraz w jaki sposób wyniki były uzyskiwane

Też fajnie np. blokowo zaprezentować na czym atak polegał

- robić też testy porównawcze z zabezpieczeniami i bez nich

- opisywać wykresy

- wnioski: który system bezpieczniejszy, która opcja bezpieczniejsza – VM czy bare-metal

- zapora sieciowa na sieciach publicznych i prywatnych

- oprócz sieciowych jakiś malware

- każdy rozdział poświęcony innemu typowi testów

1. Interfejs wlan0 w systemach Linux odnosi się do połączenia bezprzewodowego i może być używany do analizy ruchu sieciowego, jeśli działa w trybie monitorowania. [↑](#footnote-ref-1)
2. Promiscius to tryb pracy interfejsu sieciowego, w którym przechwytywane są wszystkie pakiety przechodzące przez sieć. [↑](#footnote-ref-2)
3. Podana strona umożliwia przeprowadzanie testów bezpieczeństwa, poprzez przechwytywanie danych logowania przesływanych przez http. [↑](#footnote-ref-3)