Przedmowa

Napastnicy stale biorą na cel sieci rozmaitych organizacji, w tym sklepów, banków czy firm technologicznych. W czasie, gdy przygotowywałem to wydanie Oceny bezpieczeństwa sieci, zapotrzebowanie na umiejętności reagowania na incydenty gwałtownie rosło. Choć producenci oprogramowania ciężko pracowali nad poprawieniem bezpieczeństwa swoich produktów w minionej dekadzie, złożoność systemów, a tym samym powierzchnia ataku znacznie wzrosła i mimo ich starań ogólny poziom bezpieczeństwa w Internecie się obniżył.

Taktyki napastników stały się bardziej wyrafinowane, łącząc zawiłe sposoby wykorzystania defektów oprogramowania, socjotechnikę oraz fizyczne metody docierania do wartościowych zasobów. Co jeszcze pogarsza sytuację, wiele technologii wprowadzonych w celu zabezpieczenia sieci okazało się w praktyce nieskutecznych. Tavis Ormandy, członek zespołu Google Project Zero[1], ujawnił wiele poważnych, możliwych do zdalnego wykorzystania podatności w rozmaitych produktach zabezpeiczeń[2].

Stawki rosną, a wraz z nimi wartość wyników prowadzonych badań. Badacze zabezpieczeń są zachęcani finansowo do ujawniania podatności *zero-day* osobom trzecim i pośrednikom, którzy następnie przekazują odkrycia swoim klientom, a w niektórych przypadkach odpowiedzialnie powiadamiają wytwórcę produktu. Istnieje jednak luka pomiędzy liczbą znanych powszechnie poważnych defektów oprogramowania a liczbą

1 Dobry przegląd Project Zero zawiera artykuł Andy'ego Greenberga, „Meet Project Zero, Google's Secret Team of Bug-Hunting Hackers", Wired, 15 lipca 2014 (*https://www.wired.com/2014/07/google-project-zero/*).

2 Dla przykładu Ormandy opisał podatności obecne w produktach FireEye i Sophos w artykułach „FireEye Exploitation: Project Zero's Vulnerability of the Beast" (*http://googleprojectzero.blogspot.com/2015/12/fireeye-exploitation-project-zeros.html*) oraz „Sophail: Applied Attacks Against Sophos Antivirus" (*http://lock.cmpxchg8b.com/sophailv2.pdf*). Opisał też na Twitterze exploit dla oprogramowania Kaspersky i wskazał problemy w antywirusach firm Symantec i Trend Micro.

tych, które dostępne są tylko dla uprzywilejowanych grup (np. rządów lub zorganizowanej przestępczości) i powiększa się ona każdego dnia.

Odruchową reakcją jest żądanie ścigania hackerów i powstrzymanie rozpowszechniania ich narzędzi. Jednak napastnicy, z którymi musimy się mierzyć, wraz w wybieranymi przez nich taktykami, nie są niczym innym, niż *symptomem* poważnego problemu: produkty, których używamy, nie spełniają swego przeznaczenia. Bezpieczeństwo produktu jest czymś drugorzędnym dla wielu firm technologicznych i każde wyzwanie, przed którym stajemy, jest manifestacją tego zjawiska.

Aby jeszcze pogorszyć sprawy, rządy zmilitaryzowały Internet i zrujnowały jakość kryptosystemów używanych do ochrony danych[3]. Jako profesjonaliści zabezpieczeń, musimy zachęcać do *pogłębionej obrony* w celu ograniczenia ryzyka, które zawsze będzie istniało, oraz ciężko pracować, aby zapewnić, że nasze sieci są bezpiecznym miejscem prowadzenia interesów, przechowywania danych i komunikowania się ze sobą. Życie nas wszystkich byłoby zupełnie inne bez Internetu i wolności, którą oferuje.

Przegląd

Książka poświęcona jest tylko jednemu obszarowi bezpieczeństwa komputerowego, ale szczegółowo – podejmowania testów penetracyjnych sieci w sposób strukturalny. Prezentowana metodologia pokazuje, jak zdeterminowani napastnicy przeszukują dostępne w Internetcie sieci w poszukiwaniu podatnych komponentów i jak powinniśmy wykonać analogiczne próby, aby móc ocenić jakość swojego środowiska.

Ocena bezpieczeństwa jest pierwszym krokiem, który każda organizacja musi wykonać, aby móc zarządzać ryzykiem. Poprzez przetestowanie sieci w taki sam sposób, w jaki zrobiłby to zdeterminowany napastnik, możemy zawczasu wykryć występujące w niej słabości. W tej książce łączę treści pokazujące techniki ataku z listami kontrolnymi, pokazującymi odpowiednie przeciwdziałania. W ten sposób można znaleźć właściwą strategię i odpowiednio ufortyfikować nasze środowisko.

Audytorium

Zakładam, że Czytelnik jest dobrze zaznajomiony z protokołami sieciowymi i administrowaniem systemami operacyjnymi opartymi na Unix. Jeśli jest doświadczonym inżynierem sieciowym lub konsultantem zabezpieczeń, powinien czuć się swobodnie podczas lektury poszczególnych rozdziałów. By uzyskać jak najwięcej korzyści z tej książki, czytelnik powinien znać:

3 Patrz artykuł Daniela J. Bernsteina „Making Sure Crypto Stays Insecure" (*http://cr.yp.to/talks/2014.10.18/slides-djb-20141018-a4.pdf*) oraz wykład Matthew Greena na TEDx, „Why the NSA Is Breaking Our Encryption—And Why We Should Care" (*https://youtu.be/M6qoJNLIoJI*).

- Działanie sieci na poziomie warstwy 2 modelu OSI (przede wszystkim ARP oraz znakowanie 802.1Q VLAN).
- Zbiór protokołów IPv4, w tym TCP, UDP i ICMP.
- Działanie popularnych protokołów sieciowych (np. FTP, SMTP i HTTP).
- Podstawy układu pamięci w czasie wykonywania programu oraz rejestry procesorów rodziny Intel x86.
- Zasady kryptograficzne (np. wymiany kluczy Diffie-Hellmana i RSA).
- Typowe słabości aplikacji web (XSS, CSRF, iniekcja poleceń itp.)
- Co najmniej jeden uniksopodobny system operacyjny, taki jak Linux lub Mac OS X.
- Narzędzia konfiguracyjne i kompilatory używane w środowisku Unix.

Organizacja książki

Książka ta zawiera 15 rozdziałów i trzy dodatki. Na końcu każdego rozdziału jest lista kontrolna, podsumowująca opisane zagrożenia i techniki, wraz z zalecanymi środkami zaradczymi. Dodatki zawierają materiał referencyjny, w tym listy często używanych portów TCP i UDP, które możemy napotkać podczas testów. Oto krótkie omówienie zawartości każdego rozdziału i dodatku:

- Rozdział 1, „Ocena bezpieczeństwa sieci", zawiera uzasadnienie działań związanych z badaniem bezpieczeństwa i wprowadza koncepcję zabezpieczania informacji jako procesu, a nie produktu.
- Rozdział 2, „Przebieg oceny i narzędzia", przedstawia narzędzia budujące platformę ataku używaną przez profesjonalnego konsultanta zabezpieczeń oraz taktyki testów, które należy zastosować.
- Rozdział 3, „Luki i przeciwnicy", kategoryzuje podatności oprogramowania według taksonomii, wraz z ogólnym omówieniem klas podatności i typów napastników.
- Rozdział 4, „Odkrywanie sieci z Internetu", opisuje taktyki, które potencjalny napastnik może wykorzystać do stworzenia mapy naszej sieci – od przeszukiwania otwartej sieci, po sprawdzanie DNS i odpytywanie serwerów pocztowych.
- Rozdział 5, „Lokalne odkrywanie sieci", pokazuje kroki podejmowane w celu odkrycia struktury sieci lokalnej i podsłuchiwania, wraz z metodami omijania mechanizmów zabezpieczeń 802.1Q i 802.1X.
- Rozdział 6, „Skanowanie sieci IP", omawia popularne techniki skanowania sieci i odpowiadające im aplikacje. Wylicza również narzędzia, które obsługują skanowanie każdego typu. Zawiera również opis technik unikania IDS oraz techniki niskopoziomowej analizy pakietów.

- Rozdział 7, „Ocena typowych usług sieciowych”, zajmuje się metodami testowania usług uruchamianych na różnych platformach. Omówione w tym rozdziale protokoły obejmują SSH, FTP, Kerberos, SNMP i VNC.
- Rozdział 8, „Testowanie usług firmy Microsoft”, omawia testowanie usług firmy Microsoft występujących w środowiskach korporacyjnych (NetBIOS, SMB Direct, RPC i RDP).
- Rozdział 9, „Ocena usług pocztowych”, obejmuje szczegóły oceny usług SMTP, POP3 i IMAP, przekazujących pocztę. Często usługi te okazują się podatne na wyciek informacji i ataki siłowe, a w niektórych przypadkach również na zdalne wykonanie kodu.
- Rozdział 10, „Ocena usług VPN”, zajmuje się testami usług IPsec i PPTP używanych do zapewnienia bezpiecznego dostępu do sieci i poufności danych w trakcie przesyłania.
- Rozdział 11, „Ocena usług TLS”, opisuje testowanie protokołów TLS i funkcji zapewniających bezpieczny dostęp do Web, poczty i innych usług sieciowych.
- Rozdział 12, „Architektura aplikacji Web”, definiuje elementy serwera aplikacji Web i opisuje sposoby interakcji pomiędzy nimi, w tym protokoły i formaty danych.
- Rozdział 13, „Ocena serwerów Web” zawiera techniki oceny oprogramowania serwerów Web, w tym Microsoft IIS, Apache HTTP Server oraz Nginx.
- Rozdział 14, „Ocena platform aplikacji Web”, zajmuje się szczegółami taktyk używanych do odkrywania luk w różnych platformach, takich jak Apache Struts, Rails, Django, Microsoft ASP.NET i PHP.
- Rozdział 15, „Ocena magazynów danych”, omawia zdalną ocenę serwerów bazodanowych (takich jak Oracle Database, Microsoft SQL Server lub MySQL), protokoły magazynowe oraz rozproszone magazyny typu klucz-wartości, występujące w większych systemach.
- Dodatek A, „Dobrze znane porty i typy komunikatów”, zawiera zestawienie typowych portów TCP, UDP i ICMP wraz z odsyłaczami do odpowiednich rozdziałów książki.
- Dodatek B, „Źródła informacji o podatnościach” wylicza publiczne źródła informacji o podatnościach i exploitach. Zdecydowanie nie jest to kompletna lista, ale może stanowić dobry punkt wyjścia do samodzielnego wyszukiwania źródeł.
- Dodatek C, „Niebezpieczne pakiety kryptograficzne TLS” prezentuje podatne pakiety kryptograficzne obsługiwane przez TLS, które powinny być wyłączone i których stosowania należy unikać, jeśli tylko jest to możliwe.

Korzystanie z odsyłaczy RFC i CVE

W całej książce można znaleźć liczne odsyłacze do konkretnych szkiców IETF Request for Comments (RFC)[4], dokumentów oraz wpisów bazy danych MITRE Common Vulnerabilities and Exposures (CVE)[5]. Opublikowane szkice RFC definiują wewnętrzne działanie i mechanikę protokołów, takich jak SMTP, FTP, TLS, HTTP lub IKE. Lista MITRE CVE jest zestawieniem publicznie znanej wiedzy o defektach (podatnościach) zabezpieczeń oprogramowania, zaś poszczególne wpisy (opisane poprzez rok i unikatowy identyfikator) pozwalają śledzić określone luki.

Podatności omówione w tej książce

Książka ta opisuje podatności, które mogą zostać wykorzystane zarówno przez uwierzytelnionych, jak i nieuwierzytelnionych użytkowników, nakierowane na usługi sieciowe. Przykłady taktyk, które zasadniczo leżą poza tematem tej książki, obejmują lokalną eskalację uprawnień, wywoływanie odmowy usługi (DoS) oraz włamania wykonywane przy dostępie do sieci lokalnej (w tym ataki typu man-in-the-middle).

Podatności, których odsyłacze CVE są datowane na rok 2008 i wcześniejsze, nie zostały omówione w tej książce. Wcześniejsze wydania zostały opublikowane w latach 2004 i 2007; zawierają one omówienie starszych podatności pakietów serwerowych, w tym Microsoft IIS, Apache i OpenSSL.

Ze względu na przejrzystość wiele mniej popularnych pakietów serwerowych zostało pominiętych. Jeśli taka mniej znana usługa zostanie wykryta w czasie testów, zalecane jest ręczne przeszukanie bazy danych NIST National Vulnerability Database (NVD)[6], aby poznać ich znane problemy.

Uznawane standardy oceny

Książka ta została napisana zgodnie z uznawanymi standardami testów penetracyjnych, w tym NIST SP 800-115, NSA IAM, CESG CHECK, CREST, Tiger Scheme, The Cyber Scheme, PCI DSS oraz PTES. Materiału zawartego w książce można użyć podczas przygotowywania do egzaminów z zakresu testów infrastruktury i aplikacji Web, prowadzonych przez te ciała akredytujące.

4 *http://www.ietf.org/rfc.html*

5 *https://cve.mitre.org/*

6 Wyszukiwanie według słów kluczowych: *http://web.nvd.nist.gov/view/vuln/search*

NIST SP 800-115

W roku 2008 National Institute of Standards and Technology (NIST) opublikował specjalną publikację 800-115[7], stanowiącą przewodnik techniczny testów zabezpieczeń. SP 800-115 opisuje proces oceny na wysokim poziomie, wraz z niskopoziomowymi testami, które należy podjąć wobec indywidualnych systemów.

NSA IAM

Narodowa Agencja Bezpieczeństwa USA (National Security Agency – NSA) opublikowała platformę *INFOSEC Assessment Methodology* (IAM) jako pomoc dla konsultantów i profesjonalistów zabezpieczeń spoza NSA w wykonywaniu ocen dla ich klientów. Platforma IAM definiuje trzy poziomy oceny odnoszące się do testowania sieci komputerowych:

Oszacowanie (poziom 1)
: Poziom ten obejmuje kooperacyjne (wspólnie z zainteresowaną firmą) wysokopoziomowe poznawanie docelowej organizacji, obejmujące zasady, procedury i szczegóły przepływu informacji. Na tym poziomie nie są wykonywane praktyczne testy sieci ani poszczególnych systemów.

Ocena (poziom 2)
: Ocena to wykonywany praktycznie, kooperacyjny proces, obejmujący skanowanie sieci, wykorzystanie narzędzi penetracyjnych i aplikacji o specjalnym zastosowaniu.

Red Team (poziom 3)
: Ocena typu *red team* (ang. „*czerwony zespół*") jest zewnętrznym, niekooperacyjnym testem sieci docelowej, obejmującym testy penetracyjne symulujące faktycznego napastnika. Ocena *Red team* obejmuje pełny zakres możliwych podatności.

Książka ta opisuje skanowanie podatności i techniki testów penetracyjnych używane na poziomach 2 i 3 platformy IAM.

CESG CHECK

Brytyjska Centrala Łączności Rządowej (Government Communications Headquarters – GCHQ) zawiera sekcję bezpieczeństwa informatycznego, znaną jako Communications and Electronics Security Group (CESG). Podobnie jak platforma IAM utworzona przez NSA pomaga działać konsultantom spoza instytucji rządowych zapewniać usługi testowe i doradcze, CESG prowadzi program znany jako CHECK[8] w celu oceniania i akredytacji zespołów testowych w Zjednoczonym Królestwie.

---

7 Karen Scarfone i in., „Technical Guide to Information Security Testing and Assessment", National Institute of Standards and Technology, wrzesień 2008 (*http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf*).

8 „CHECK Fundamental Principles", National Cyber Security Centre, 23 października 2015 (*https://www.cesg.gov.uk/articles/check-fundamental-principles*

W odróżnieniu od platformy IAM, która obejmuje wiele aspektów bezpieczeństwa informatycznego (w tym przegląd zasad, sprawdzenie antywirusów, kopii zapasowych i planu przywracania po awariach), CHECK zasadniczo ogranicza się do oceny bezpieczeństwa sieciowego. Istnieje także drugi program o nazwie CESG Listed Adviser Scheme (CLAS), który zajmuje się bezpieczeństwem informacji w szerszym znaczeniu zgodnie z zaleceniami standardu ISO/IEC 27001, a także tworzeniem zasad zabezpieczeń i inspekcją.

W celu zademonstrowania swoich umiejętności i uzyskania akredytacji konsultanci przechodzą zaaprobowane przez CESG szkolenia (głównie te organizowane przez CREST i Tiger Scheme). CHECK odnotowuje następujące umiejętności jako przykłady kompetencji technicznej:

- Wykorzystanie narzędzi odczytywania informacji DNS dla pojedynczych rekordów i całych zbiorów, w tym rozumienie struktur rekordów DNS powiązanych z wyszukiwaniem hostów.
- Użycie narzędzi mapowania i próbkowania sieci ICMP, TCP i UDP.
- Zademonstrowanie przechwytywania banera usługi TCP.
- Odczytywanie informacji przy użyciu SNMP, w tym znajomość struktury MIB odnoszącej się do konfiguracji docelowego systemu i tras sieciowych.
- Znajomość typowych słabości routerów i przełączników dotyczących dostępu za pośrednictwem protokołów Telnet, HTTP, SNMP i TFTP i konfiguracji.

Poniższe umiejętności są specyficzne dla systemów Unix:

- Wyliczanie użytkowników (via *finger*, *rusers*, *rwho* oraz techniki SMTP).
- Wyliczanie usług RPC i zademonstrowanie powiązanych z nimi konsekwencji dotyczących zabezpieczeń.
- Identyfikacja słabości Network File System (NFS).
- Testowanie słabości w *r*-serwisach (*rsh*, *rexec* i *rlogin*).
- Wykrywanie niezabezpieczonych serwerów X Windows.
- Identyfikacja słabości w usługach Web, FTP i Samba.

Kompetencje specyficzna dla Windows obejmują:

- Sprawdzenie usług NetBIOS, SMB i RPC w celu wyliczania użytkowników, grup, udziałów sieciowych, domen, kontrolerów domeny, zasad haseł i powiązanych z nimi słabości.
- Wydobywanie nazw użytkowników i haseł za pośrednictwem usług SMB i RPC.
- Zademonstrowanie obecności znanych luk w Microsoft IIS i SQL Server.

W tej książce udokumentowane są taktyki oceny związane z powyższymi zagadnieniami, wraz z pomocniczymi informacjami, które pozwalają na dogłębne zrozumienie możliwych podatności. Choć program CHECK obejmuje metodologię, którą powinni posługiwać się

konsultanci pragnący pracować dla rządu Zjednoczonego Królestwa, również w innych miejscach dobrze jest znać te wymagania.

Kwalifikacje uznawane przez CESG

W Wielkiej Brytanii działa kilka organizacji, które prowadzą szkolenia i egzaminy zaaprobowane przez CESG. Certyfikacje udzielane przez te organizacje są uznawane przez CESG jako równoważne CHECK:

CREST (*http://www.crest-approved.org/*)
: CREST jest organizacją nonprofit, która reguluje branżę testów penetracyjnych poprzez przyznawanie akredytacji w postaci programów *certyfikowanych testerów infrastruktury* oraz *certyfikowanych testerów aplikacji web*. Poprzez partnerstwo z CESG kwalifikacje testerów certyfikowane przez CREST są uznawane za wymagane dla szefów zespołów CHECK, a tym samym wiele organizacji używa tej podstawy programowej przy szkoleniu swoich zespołów testowych.

Tiger Scheme (*http://www.tigerscheme.org/*)
: Drugim ciałem egzaminacyjnym, które współpracuje z instytucjami rządowymi i branżowymi, jest Tiger Scheme. Dostępne są trzy poziomy akredytacji: *associate* (współpracownik), *qualified* (dyplomowany) oraz *senior*, które są uznawane przez CESG i mogą zostać użyte w celu zapewnienia właściwego statusu członków zespołu i liderów CHECK.

The Cyber Scheme (*http://www.thecyberscheme.co.uk/*)
: Certyfikacja Cyber Scheme Team Member (CSTM) jest uznawana przez CESG jako równoważna wymaganiom na członka zespołu CHECK. Organizacja prowadzi zarówno szkolenia, jak i egzaminy poprzez swoich zaaprobowanych partnerów.

PCI DSS

Rada standardów zabezpieczeń branży kart płatniczych (Payment Card Industry Security Standards Council – PCI SSC) utrzymuje standard bezpieczeństwa danych PCI DSS, który wymaga od instytucji przetwarzających płatności, sprzedawców i inne jednostki używające danych kart płatniczych stosowania się do określonych celów kontrolnych (*control objectives*), w tym:

- Budowanie i utrzymywanie zabezpieczonej sieci.
- Ochrona danych właścicieli kart.
- Utrzymywanie programu zarządzania podatnościami.
- Implementacja silnych środków kontroli dostępu.
- Regularne monitorowanie i testowanie sieci.
- Utrzymywanie zasad bezpieczeństwa informacji.

Obecnie obowiązującym standardem jest PCI DSS w wersji 3.1. W tym dokumencie występują dwa wymagania skanowania pod kątem podatności i testów penetracyjnych, dotyczące firm przetwarzających płatności oraz handlowców:

Wymaganie 11.2
: Nakazuje kwartalne skanowanie pod kątem podatności, zarówno wewnętrzne, jak i zewnętrzne. Wykonanie zewnętrznego testu musi wykonać firma posiadająca certyfikat PCI SSC Approved Scanning Vendor (ASV), choć akredytacja ta nie jest wymagana dla testów wewnętrznych.

Wymaganie 11.3
: Wymaga przeprowadzenia corocznych wewnętrznych i zewnętrznych testów penetracyjnych, wykonywanych przez kwalifikującą się jednostkę, zgodnie z ogólnie akceptowanymi najlepszymi praktykami (czyli NIST SP 800-115).

Książka ta jest zgodna z wymaganiami NIST SP 800-115 i innymi opublikowanymi standardami, zatem można wykorzystać przedstawioną w niej metodologię przy wykonywaniu wewnętrznych i zewnętrznych testów w celu wypełnienia wymagania 11.3.

PTES

PCI SSC uznaje *Penetration Testing Execution Standard* (PTES)[9] za platformę odniesienia dla testów; składa się ona z siedmiu sekcji. Witryna PTES zawiera szczegółowe omówienie poszczególnych sekcji, jak poniżej:

- Interakcje wstępne (przed zaangażowaniem).
- Gromadzenie danych wywiadowczych.
- Modelowanie zagrożeń.
- Analiza podatności.
- Wykorzystanie podatności.
- Działania po wykorzystaniu podatności.
- Raportowanie.

Witryna mirroru dla narzędzi wymienionych w książce

W całej książce podawane są adresy URL dla używanych narzędzi, dzięki czemu można wyszukać najnowsze wersje plików i dokumentów z odpowiednich źródeł. Jeśli Czytelnik obawia się obecności koni trojańskich czy innej złośliwej zawartości, mogę jedynie zapewnić, że wszystkie zostały sprawdzone i są dostępne poprzez witrynę powiązaną z książką[10].

9 Szczegóły tego standardu zawiera strona *http://www.pentest-standard.org/*

10 *http://examples.oreilly.com/networksa/tools/*

Bardzo możliwe, że przy próbie dostępu do tej strony pojawi się ostrzeżenie o zagrożeniach – ostatecznie, wszystko to są narzędzia hackerskie!

Korzystanie z przykładów kodu

Materiały uzupełniające (przykłady kodu, ćwiczenia i tak dalej) są dostępne do pobrania ze strony powiązanej z tą książką pod adresem *http://examples.oreilly.com/9780596006112/tools/*.

Książka ta ma pomóc Czytelnikowi w wykonaniu pracy. W ogólności kodu zawartego w książce można użyć w swoich programach i dokumentacji. Nie trzeba kontaktować się z nami w celu uzyskania zgody, o ile nie zamierza się zreprodukować znaczącej części kodu. Dla przykładu, napisanie programu, który wykorzysta wiele fragmentów kodu z tej książki nie wymaga zezwolenia. Sprzedawanie lub dystrybucja przykładów (na przykład na CD-ROM) przykładów z książek O'Reilly wymaga takiej zgody. Odpowiadanie na pytanie poprzez zacytowanie tej książki i dołączenie przykładu nie wymaga zezwolenia. Włączenie znaczącej ilości kodu przykładowego z tej książki do dokumentacji wymaga zezwolenia.

Będziemy wdzięczni za wskazanie źródła, choć nie jest to wymagane. Przypis powinien zwykle zawierać nazwisko autora, tytuł, wydawnictwo oraz numer ISBN. Na przykład: „Chris McNab, Ocena bezpieczeństwa sieci, O'Reilly/APN Promise, Copyright 2016 Chris McNab, 978-83-7541-212-3".

W przypadku wątpliwości, czy planowane zastosowanie przykładowego kodu wykracza poza przedstawione powyżej zezwolenia, prosimy o skontaktowanie się z nami za pośrednictwem adresu e-mail: *permissions@oreilly.com*.

Konwencje użyte w tej książce

Oto konwencje typograficzne przyjęte w tej książce:

Zwykły tekst
: Odnosi się do elementów, tytułów i opcji menu oraz klawiszy skrótów (takich jak Alt czy Ctrl).

Kursywa
: Wyróżnia nowe terminy, adresy URL, adresy email, ścieżki, nazwy i rozszerzenia plików.

`Czcionka stałopozycyjna`
: Służy do przedstawiania fragmentów kodu programu, a także do odwoływania się w tekście do elementów programu, takich jak nazwy zmiennych, funkcji i baz danych, typy danych, zmienne środowiskowe, instrukcje czy słowa kluczowe.

`Czcionka stałopozycyjna wytłuszczona`
: Wskazuje tekst wpisywany przez użytkownika.

Ta ikona oznacza wskazówkę lub sugestię.

Ta ikona oznacza uwagę ogólną.

Ta ikona wskazuje ostrzeżenie o niebezpieczeństwie.

Podziękowania

W ciągu mojej kariery wiele osób zapewniło mi nieocenioną pomoc i większość doskonale wie, jak wiele dla mnie znaczy. Mój zmarły przyjaciel Barnaby Jack pomógł mi w 2009 roku zdobyć pracę, która zmieniła moje życie na lepsze. Bardzo mi go brakuje i Jägermeister nie smakuje już tak samo.

Ze względu na typ osobowości INTP zwykle jestem milczący i niekiedy trudno ze mną wytrzymać. Nie oznacza to, że chcę być niemiły i naprawdę doceniam tych, którzy mi towarzyszą od lat, a szczególnie moje przyjaciółki i rodzinę.

Chciałbym też podziękować zespołowi O'Reilly Media za nieustające wsparcie, cierpliwość i niewzruszone zaufanie. Stworzenie tej książki nie było łatwe, ale dzięki ich pomocy mogę mieć nadzieję, że pozwoli ona uczynić ten świat nieco bezpieczniejszym miejscem.

Recenzenci techniczni i współautorzy

Systemy komputerowe stały się tak mgliste, że musiałem w wielu przypadkach odwoływać się do ekspertów w określonej dziedzinie, aby móc omówić problemy dotyczące różnych technologii. Zebranie tego materiału razem zwyczajnie nie byłoby możliwe bez pomocy, której udzieliły mi następujące utalentowane osoby: Car Bauer, Michael Collins, Daniel Cuthbert, Benjamin Delpy, David Fitzgerald, Rob Fuller, Chris Gates, Dane Goodwin, Robert Hurlbut, David Litchfield, HD Moore, Ivan Ristić, Tom Ritter, Andrew Ruef i Frank Thornton.

Rozdział 1

Ocena bezpieczeństwa sieci

Rozdział ten stanowi wprowadzenie do ekonomicznych reguł leżących u podłoża eksploatowania sieci komputerowych oraz ich obrony: opisuje zarówno bieżący stan, jak i zmiany, które nastąpiły w ciągu ostatniej dekady. Aby móc zrealizować środowiska nadające się do obronienia, konieczne jest przyjęcie proaktywnej postawy wobec bezpieczeństwa – podejścia rozpoczynającego się od rzetelnej oceny, pozwalającej zidentyfikować nasze słabości. Istnieje wiele różnych technik prowadzenia takiej oceny, od statycznej analizy określonej aplikacji i jej kodu po dynamiczne testy wdrożonych już i działających systemów. W kolejnych podrozdziałach opiszę i sklasyfikuję każdą z dostępnych opcji testowania, a na koniec wyliczę te dziedziny, które zostaną szczegółowo omówione w dalszych rozdziałach tej książki.

Stan wiedzy

Pracę nad pierwszym wydaniem tej książki rozpocząłem blisko dwadzieścia lat temu, długo wcześniej, nim nadużycia sieci komputerowych poprzez działania rządów i przestępczości zorganizowanej rozwinęły się do skali, którą znamy dziś. Biznes wyszukiwania i sprzedawania exploitów *zero-day* dopiero się tworzył, zaś hackerzy byli szczytowymi drapieżnikami świata online, handlującymi warezami za pośrednictwem IRC.

Bieżący stan spraw budzi głęboki niepokój. Dzisiejszy sposób życia w wielkim stopniu uzależniony jest od sieci komputerowych i aplikacji, które już są bardzo złożone, przy czym złożoność ta rozwija się w wielu kierunkach (można tu pomyśleć o aplikacjach chmurowych, urządzeniach medycznych, inteligentnych samochodach, samolotach czy systemach wbudowanych). Rosnące wykorzystywanie potencjalnie podatnych produktów w nieunikniony sposób wprowadza słabości.

Internet jest podstawowym mechanizmem umożliwiającym funkcjonowanie zglobalizowanego systemu gospodarczego i opiera się na nim już niemal wszystko, w tym handel, logistyka łańcuchów zaopatrzenia, praca zespołowa, zarządzanie krytyczną infrastrukturą, relacje z wydarzeń czy wsparcie dla produktów. Studium *International Institute for Applied Systems Analysis* (IIASA)[1] przewiduje, że całkowita utrata usług internetowych w skali państwa doprowadzi do przerwania działania zaopatrzenia w żywność w ciągu zaledwie trzech dni.

Szerokie rozpowszechnienie technologii znacząco zwiększa negatywne skutki gospodarcze postrzegane przez ofiary ataków komputerowych. Przykładem może być użycie worma Stuxnet[2] przeciwko irańskiej instalacji wzbogacania uranu w Natanz (spowodowało redukcję możliwości produkcyjnych o 30% na okres kilku miesięcy). W przypadku saudyjskiego koncernu Aramco atak malware[3] w roku 2012 spowodował dziesięciodniowe wyłączenie wszystkich operacji. Paraliżujące ataki trwały nadal w latach 2015 i 2016 – przykładem mogą być Sony Pictures Entertainment i Hollywood Presbyterian Medical Center[4].

Istnieje rażąca luka pomiędzy pomysłowymi i bogato wyposażonymi przeciwnikami a tymi, którzy muszą zapewnić ochronę swoich sieci komputerowych. Dzięki rosnącej mobilności pracowników, rozpowszechniania technologii bezprzewodowych i usług chmurowych systemy nawet tych organizacji, które są najbardziej świadome problemów zabezpieczeń, takich jak Facebook[5], mogą zostać opanowane i przejęte. Aby jeszcze pogorszyć sprawę, krytyczne luki pozstawały długo niewykryte w samych technologiach i bibliotekach programowych używanych do zapewnienia bezpieczeństwa. Wystarczy wymienić najgłośniejsze przykłady:

- RSA BSAFE, używający domyślnie podatnego algorytmu Dual_EC_DRBG[6].
- Wyciek informacji powodowany przez rozszerzenie heartbeat OpenSSL 1.0.1[7].

Sieci możliwe do obrony naturalnie istnieją, ale zwykle są to niewielkie, silnie zabezpieczone enklawy o wysokich standardach, budowane przez organizacje przemysłu obronnego. Enklawy takie są ściśle skonfigurowane i monitorowane; uruchamiane w nich są tylko

1 Leena Ilmola-Sheppard i John Casti, „Case Study: Seven Shocks and Finland", Innovation and Supply Chain Management 7, nr 3 (2013): 112–124 (*http://pure.iiasa.ac.at/10111/*)

2 Kim Zetter, „An Unprecedented Look at Stuxnet, the World's First Digital Weapon", Wired, 3 listopada 2014 (*http://bit.ly/2aNyLq1*).

3 John Leyden, „Hack on Saudi Aramco Hit 30,000 Workstations, Oil Firm Admits", The Register, 29 sierpnia 2012 (*http://bit.ly/2aNyJhW*).

4 Peter Elkins, „Inside the Hack of the Century", Fortune.com, 25 czerwca 2015 (*http://fortune.com/* sony-hack-part-1/); Robert Mclean, „Hospital Pays Bitcoin Ransom After Malware Attack", CNN Money, 17 lutego 2016 (*http://cnnmon.ie/2aNyP9e*).

5 Orange Tsai, „How I Hacked Facebook, and Found Someone's Backdoor Script", DEVCORE Blog, 21 kwietnia 2016 (*http://bit.ly/2aNyTpj*).

6 Joseph Menn, „Secret Contract Tied NSA and Security Industry Pioneer", Reuters, 20 grudnia 2013 (*http://reut.rs/2aNzqHU*).

7 CVE-2014-0160

zaufane i przetestowane systemy operacyjne oraz przebadane oprogramowanie. Wiele z nich zawiera również wyspecjalizowany sprzęt wymuszający jednokierunkową komunikację pomiędzy określonymi komponentami. Takie wzmocnione środowiska nie są „niewzruszalne", ale mogą być skutecznie obronione przed atakami większości rodzajów dzięki monitorowaniu i szybkiej reakcji.

Sieci, które są najbardziej narażone, to te, które zawierają największą liczbę elementów. Liczne punkty wejściowe zwiększają potencjalne możliwości włamania, a zarządzanie ryzykiem staje się trudne. Czynniki te składają się na *dylemat obrońcy* – podczas gdy obrońcy muszą zagwarantować integralność całego systemu, napastnik potrzebuje tylko wykorzystać pojedynczą słabość.

Zagrożenia i powierzchnia ataku

Napastnicy aktywnie atakujący systemy komputerowe rozciągają się od organizacji finansowanych przez państwa, poprzez zorganizowaną przestępczość, aż po entuzjastycznych amatorów. Dzięki dysponowaniu większymi zasobami, niż wszystkie pozostałe grupy łącznie, sponsorowani przez państwa napastnicy są obecnie czołowym zagrożeniem.

Dzięki zrozumieniu, jaka jest potencjalna powierzchnia ataku, możemy próbować ilościowo określić ryzyko. Narażone obszary większości przedsiębiorstw obejmują systemy klienckie (komputery biurkowe, laptopy i urządzenia mobilne), serwery internetowe, aplikacje chmurowe oraz komponenty infrastruktury (bramy VPN, routery i zapory ogniowe).

Atakowanie oprogramowania klienckiego

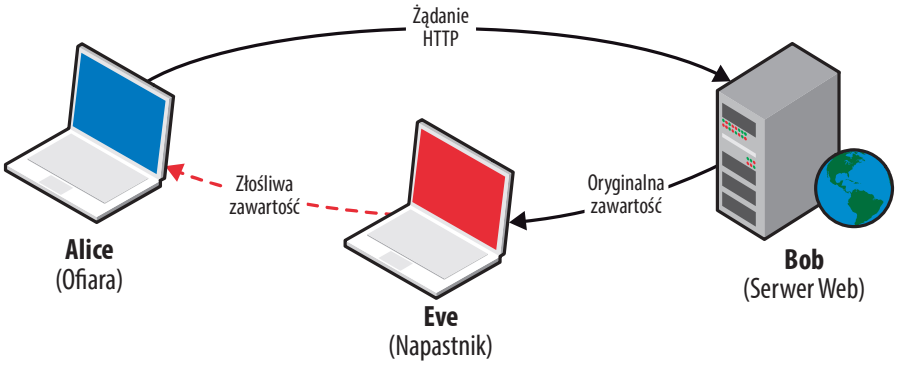
Aplikacje biurkowe i pakiety oprogramowania klienckiego (np. Microsoft Office, przeglądarki Web, w tym Google Chrome, oraz narzędzia administracyjne, takie jak PuTTY) mogą zostać bezpośrednio zaatakowane przez napastnika dysponującego dostępem do sieci lub niebezpośrednio poprzez przesłanie złośliwej zawartości do przetworzenia (np. odpowiednio spreparowanego pliku PDF lub arkusza Microsoft Excel).

Aby zademonstrować brak zabezpieczeń pakietów oprogramowania klienckiego, wystarczy spojrzeć na rezultaty konkursu 2014 Pwn2Own. Francuska firma VUPEN[8] zajmująca się zabezpieczeniami wygrała łącznie 400000 dolarów po skutecznym wykorzystaniu luk w Microsoft Internet Explorer 11, Adobe Reader XI, Google Chrome, Adobe Flash oraz Mozilla Firefox w 64-bitowej wersji Windows 8.1. Firma wykorzystała łącznie 11 różnych exploitów *zero-day* do osiągnięcia celów[9].

8 Firma VUPEN zakończyła działalność w roku 2015, a jej założyciele uruchomili nowe przedsięwzięcie o nazwie ZERODIUM.

9 Michael Mimoso, „VUPEN Discloses Details of Patched Firefox Pwn2Own Zero-Day", Threatpost Blog, 21 maja 2014 (*http://bit.ly/2aNz3wU*).

Napastnicy dysponujący dostępem do sieci często wykorzystują podatności przeglądarek do przejmowania zasobów o wielkiej wartości (np. kont administratorów systemów w takich organizacjach jak OPEC[10]) przy użyciu ataku typu *man-in-the-middle* – wstrzykując podstępny kod do jawnej sesji HTTP. Gdy docelowy system przetworzy tę zawartość, wklejony kod pozostaje w pamięci i udostępnia niskopoziomowy dostęp do systemu, co demonstruje rysunek 1-1.



Rysunek 1-1 *Dostęp do sieci wykorzystany w celu dostarczenia złośliwej zawartości*

Ataki takie nie są ograniczone tylko do nieszyfrowanych sesji sieciowych. Dobrze wyposażony napastnik dysponujący niskopoziomowym dostępem do sieci (np. kontrolujący router) może wykonać niewykrywalny atak MITM na uprawnioną szyfrowaną sesję (taką jak HTTPS lub SSL VPN) dzięki przejęciu klucza prywatnego, co można uzyskać albo poprzez słabość oprogramowania taką jak wyciek informacji z komunikatów heartbeat OpenSSL, albo wykorzystując niedostatki zabezpieczeń.

Atakowanie oprogramowania serwerowego

Oprogramowanie serwerowe wcale nie wypada lepiej, częściowo ze względu na rosnącą liczbę warstw abstrakcji i dołączanie nowych technologii. W roku 2013 zarówno platforma aplikacji Web Rails, jak i odwrotne proxy Nginx wykazały podatność na zdalne wykonanie kodu:

- Nginx 1.3.9 do 1.4.0 – przepełnienie stosu przez kodowanie fragmentami (*chunked encoding*)[11]
- Rails 2.3 oraz 3.x Action Pack YAML – wada deserializacji[12]

10 „Oil Espionage: How the NSA and GCHQ Spied on OPEC", SPIEGEL ONLINE, 11 listopada 2013 (*http://bit.ly/2aNz83G*).

11 CVE-2013-2028

12 CVE-2013-0156

Podatność na kodowanie fragmentami w Nginx była bardzo podobna do znalezionej przez Neela Mehta w roku 2002 w serwerze Apache[13] – demonstruje to, jak znane wady mogą pojawiać się w nowych pakietach oprogramowania, jeśli projektanci zapomną o spoglądaniu wstecz.

Atakowanie aplikacji Web

Podatności w aplikacjach Web wynikają zazwyczaj z obsługiwania dodatkowych funkcjonalności i zwiększenia wystawienia interfejsów (API) pomiędzy składnikami. Przykładem szczególnie dotkliwej klasy problemu jest przetwarzanie zewnętrznej jednostki XML (*XML eXternal Entity* – XXE), w którym złośliwy kod XML jest przedstawiany aplikacji Web, powodując zwrócenie poufnych treści. W roku 2014 badacze odkryli wiele podatności przetwarzania XXE w środowisku produkcyjnym Google. W jednym przypadku treść XML załadowano do narzędzia Google Public Data Explorer[14], definiując zewnętrzny element `payload`:

```
<!ENTITY % payload SYSTEM "file:///etc/”>
<!ENTITY % param1 '<!ENTITY &#37; internal SYSTEM "%payload;" >' >
%param1; %internal;
```

Przykład 1-1 pokazuje, jak ten XML został przetworzony po stronie serwera, ujawniając zawartość katalogu.

Przykład 1-1 *Odczytywanie plików ze środowiska produkcyjnego Google*

```
XML parsing error. Line 2, Column: 87: no protocol: bash.bashrc bashrc bashrc.google
borgattrs.d borgattrs-msv.d borgletconf.d capabilities chroots chroots.d container.d
cron cron.15minly cron.5minly cron.d cron.daily cron.hourly cron.monthly cron.weekly
crontab csh.cshrc csh.login csh.logout debian_version default dpkg fsck.d fstab
google googleCA googlekeys groff group host.conf hosts hotplug init.d inittab inputrc
ioctl.save iproute2 issue issue.net kernel lilo.conf lilo.conf.old localbabysitter.d
localbabysitter-msv.d localtime localtime.README login.defs logmanagerd logrotate.
conf logrotate.d lsb-base lsb-release magic magic.mime mail mail.rc manpath.config
mced mime.types mke2fs.conf modprobe.conf modprobe.d motd msv-configuration msv-
managed mtab noraidcheck nsswitch.conf passwd passwd.borg perfconfig prodimage-
release-notes profile protocols rc.local rc.machine rc0.d rc1.d rc2.d rc3.d rc4.d
rc5.d rc6.d rcS.d resolv.conf rpc securetty services shadow shells skel ssdtab ssh
sudoers sysconfig sysctl.conf sysctl.d syslog.d syslog-ng_configs_src syslog-ng.conf
sysstat tidylogs.d vim wgetrc
```

13 CVE-2002-0392

14 *http://examples.oreilly.com/networksa/tools/google-xxe.pdf*

Eksponowana logika

Jedną z metod poznawania potencjalnego obszaru ataku systemu komputerowego jest przeanalizowanie ujawnianej przez niego wewnętrznej logiki. Przy zdalnym dostępie tego typu logika może być częścią dostępnej z Internetu usługi sieciowej (np. obsługa kompresji lub kodowanie fragmentami w serwerze Web) lub mechanizmu przetwarzania używanego przez klienta (takiego jak renderowania PDF). Lokalnie wewnątrz systemów operacyjnych jądro systemu i sterowniki urządzeń (działające na wysokich uprawnieniach) również udostępniają logikę uruchamianym aplikacjom.

Powierzchnią ataku jest udostępniona logika (zazwyczaj uprzywilejowana), która może zostać wykorzystana przez napastnika w celu osiągnięcia jakiegoś celu – może być to ujawnienie sekretów, umożliwienie wykonania kodu lub wymuszenie odmowy usługi. W ogólności napastnicy mogą posłużyć się wystawioną logiką dwiema drogami:

- Bezpośrednio poprzez podatną funkcję wewnątrz wystawionej usługi sieciowej.
- Pośrednio, jak w przypadku parsera uruchomionego w klienckim punkcie końcowym, do którego dostarczany jest złośliwy materiał pochodzący z ataku MITM, z wiadomości email, komunikatora błyskawicznego lub innymi środkami.

Przykłady eksponowanej logiki

Dobrym przykładem jest podatność na zdalne wykonanie polecenia w powłoce GNU *bash*, znana jako *shellshock*[15]. Wiele aplikacji w systemach opartych na Unix (w tym Linux oraz Mac OS X) używają powłoki poleceń *bash* jako brokera do wykonywania niskopoziomowych operacji systemowych. W roku 1989 w powłoce tej wprowadzono podatność, która pozwalała na wykonanie określonych poleceń poprzez przesłanie do powłoki odpowiednio spreparowanych, złośliwych zmiennych środowiskowych. Luka ta została odkryta i upubliczniona dopiero po dwudziestu pięciu latach przez Stéphane Chazelas[16].

Aby móc wykorzystać podatność, napastnik musi zidentyfikować ścieżkę, którą zawartość kontrolowana przez użytkownika może zostać przekazana do podatnej powłoki. Dwa przykłady możliwych do wykorzystania zdalnie ścieżek i warunki wstępne to:

Apache HTTP Server
: Przedstawienie nagłówka User-Agent zawierającego złośliwą zmienną środowiskową skryptowi CGI wewnątrz serwera Apache (za pośrednictwem *mod_cgi*) powoduje wykonanie polecenia, co można zademonstrować w Metasploit[17].

DHCP
: Wielu klientów DHCP przekazuje i wykonuje polecenia w powłoce *bash* podczas konfigurowania interfejsów sieciowych. Poprzez skonfigurowanie oszukańczego

15 CVE-2014-6271

16 Nicole Perlroth, „Security Experts Expect 'Shellshock' Software Bug in Bash to Be Significant", New York Times, 25 września 2014 (*http://nyti.ms/299Eeml*).

17 Moduł *apache_mod_cgi_bash_env_exec* (*http://bit.ly/2aNBrDK*).

serwera DHCP, przesyłającego złośliwe zmienne środowiskowe w odpowiedziach DHCP do klientów można wykonać określone polecenia w podatnych systemach, co również demonstruje Metasploit[18].

Istota i wykorzystanie wystawionej logiki

Branża wyszukiwania podatności opiera się na wykorzystywaniu wystawionej logiki w oprogramowaniu do wykonywania użytecznego działania. Większość exploitów przeglądarek opiera się na łączeniu wielu defektów w celu pominięcia zabezpieczeń i wykonania własnego kodu.

W roku 2012 uznany hacker Pinkie Pie opracował exploit przeglądarki Google Chrome[19] używający sześciu oddzielnych luk do osiągnięcia wykonania kodu. Badacze znaleźli błędy w funkcje wstępnego rendowania Chrome, w buforze poleceń GPU, w warstwie IPC i w menedżerze rozszerzeń.

Niskopoziomowa ocena oprogramowania jest formą sztuki, w której stosunkowo niewielka liczba uzdolnionych badaczy odkrywa subtelne defekty oprogramowania i łączy je ze sobą w celu wykonania użytecznych działań. W każdym iteracyjnym kroku testowana jest potencjalna powierzchnia ataku w celu zidentyfikowania zagrożeń dla większego systemu.

Odmiany sposobów testowania

W celu zamapowania i przetestowania ścieżek wystawionej logiki systemu organizacje mogą wykorzystać rozmaite podejścia. Na dzisiejszym rynku można znaleźć wielu dostawców oferujących usługi testów statycznych i dynamicznych wraz z narzędziami analitycznymi służącymi do identyfikowania potencjalnych ryzyk i słabości.

Analiza statyczna

Audyt kodu źródłowego aplikacji, konfiguracji serwerów i infrastruktury oraz ogólnej architektury może być czasochłonny, ale jest jednym z najbardziej efektywnych sposobów identyfikowania podatności danego systemu. Wadą analizy statycznej w rozległym środowisku jest koszt uruchomienia takiego programu (głównie z powodu ogromnych rozmiarów materiałów tworzonych przez tego typu narzędzia oraz konieczności eliminowania fałszywych rozpoznań), co powoduje, że ważne jest właściwe określenie zakresu testów i odpowiednie priorytetyzowanie wysiłków.

Podejście audytu technicznego i recenzji obejmuje:

- Przegląd projektu

18 Moduł *dhclient_bash_env* (*http://bit.ly/2aNzOWI*)

19 Jorge Lucangeli Obes i Justin Schuh, „A Tale of Two Pwnies (Part I)", Chromium Blog, 22 maja 2012 (*http://bit.ly/2aNA3Bq*).

- Przegląd konfiguracji
- Statyczną analizę kodu

Mniej techniczne elementy do rozważenia mogą obejmować klasyfikowanie i etykietowanie danych, przegląd środowiska fizycznego, bezpieczeństwo personelu, a także kształcenie, szkolenia i uświadamianie pracowników.

Przegląd projektu

Analiza architektury i projektu systemu obejmuje przede wszystkim poznanie rozmieszczenia i konfiguracji mechanizmów zabezpieczeń w danym środowisku (zarówno opartych na sieci, takich jak listy kontroli dostępu, jak i niskopoziomowych mechanizmów systemowych, takich jak piaskownice), oszacowanie skuteczności tych mechanizmów oraz w razie potrzeby zaproponowanie zmian w projekcie architektury i systemu.

Common Criteria[20] to międzynarodowy standard certyfikacji bezpieczeństwa komputerowego, który można stosować wobec systemów operacyjnych, aplikacji oraz produktów zawierających deklarowane zabezpieczenia. Standard obejmuje siedem poziomów gwarancji, od EAL1 (przetestowane funkcjonalnie) poprzez EAL4 (zaprojektowane, przetestowane i zrecenzowane zgodnie z metodologią) do EAL7 (formalnie zweryfikowany projekt oraz testy). Wiele komercyjnych systemów operacyjnych jest typowo ocenianych jako EAL4, zaś systemy udostępniające wielopoziomowe zabezpieczenia otrzymują ocenę co najmniej EAL4. Istnieją też inne podobne standardy, w tym opracowane przez CESG (brytyjską agencję rządową) schematy CAPS oraz CPA (*http://bit.ly/2aNAknz*).

Formalna weryfikacja projektu lub architektury systemu może być bardzo kosztownym ćwiczeniem. Często jednak pobieżny przegląd architektury przez doświadczonego profesjonalistę w dziedzinie zabezpieczeń może wskazać potencjalne pułapki i problemy, które powinny zostać wyeliminowane, takie jak niewłaściwa segmentacja sieci lub niewystarczająca ochrona danych w trakcie przesyłania.

Przegląd konfiguracji

Niskopoziomowy audyt komponentów systemu może obejmować przegląd infrastruktury (czyli zapór ogniowych, routerów, switchy, magazynu oraz infrastruktury wirtualizacyjnej), konfiguracji systemów operacyjnych serwerów i urządzeń (np. Windows Server, Linux lub sprzętu F5) oraz konfiguracji aplikacji (takiej jak konfiguracja serwera Apache lub OpenSSL).

Różne organizacje, w tym NIST, NSA oraz DISA, udostępniają listy kontrolne oraz wskazówki konfiguracyjne dla typowych systemów operacyjnych, w tym Mac OS X, Microsoft Windows oraz Linux. Zasoby te są dostępne online:

- NIST: *http://web.nvd.nist.gov/view/ncp/repository*
- NSA: *http://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/*

20 ISO/IEC 15408

- DISA: *http://iase.disa.mil/stigs/*

Dzięki wykonaniu analizy luk względem tych standardów możliwe jest zidentyfikowanie niedociągnięć w konfiguracji systemu operacyjnego i działanie w celu zapewnienia jednolitej, wzmocnionej konfiguracji w całym środowisku. Skanery podatności (w tym Rapid7 Nexpose) zawierają zasady skanowania STIG proponowane przez DISA, dzięki czemu identyfikowanie luk jest prostolinijne, od zatwierdzonych skanów po wykonanie pogłębionych testów.

Statyczna analiza kodu

NIST oraz Wikipedia utrzymują listy narzędzi analizy kodu, jak poniżej:

- *http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html*
- *https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis*

Narzędzia takie identyfikują powszechnie spotykane niedociągnięcia w oprogramowaniu napisanym w określonych językach, takich jak C/C++, Java lub .NET. Zespół HP Fortify opublikował taksonomię[21] błędów zabezpieczeń oprogramowania, które można zidentyfikować podczas statycznej analizy kodu, w tym weryfikowanie danych wejściowych i ich reprezentowanie, naruszenie API, podatne funkcje zabezpieczeń, słabości związane z czasem lub stanem aplikacji, niedociągnięcia w obsłudze błędów oraz niską jakość kodu powodującą bugi w obsłudze pamięci. Kategorie te zostaną omówione szczegółowo w rozdziale 3.

Narzędzia statycznej analizy kodu wymagają pewnego dostrajania, aby zredukować szum w uzyskanych wynikach i skupić się na istotnych fragmentach kodu (tzn. podatnościach, które mogą być wykorzystane w praktyce). Tego rodzaju niskopoziomowa analiza jest odpowiednia dla krytycznych komponentów systemu, jako że koszt wykonania analizy i przejrzenia wyników może być znaczący.

Testowanie dynamiczne

Wystawiona logika może zostać oceniona poprzez dynamiczne testy uruchomionego systemu (w odróżnieniu od statycznej analizy kodu czy przeglądu konfiguracji komponentów systemu). Do popularnych typów testów dynamicznych należą:

- Testowanie infrastruktury sieci.
- Testowanie aplikacji Web.
- Testowanie usług Web (np. udostępnianych API dla aplikacji mobilnych).
- Socjotechniki opartej na Internecie.

21 *https://vulncat.hpefod.com/*

Jako że testowanie jest wykonywane z perspektywy napastnika (takiego jak nieuwierzytelniony włamywacz z sieci, uwierzytelniony użytkownik aplikacji lub klient mobilny), odkrycia są często związane z faktycznie istniejącymi (i dającymi się wykorzystać) zagrożeniami dla systemu.

Testowanie infrastruktury sieci

Przy użyciu narzędzi skanujących (np. Nmap, Nessus, Rapid7 Nexpose czy QualysGuard) można wykonać mapę sieciowej powierzchni ataku i ocenić ją pod kątem znanych podatności. Następnie wykonuje się ręczne analizy w celu lepszego zbadania powierzchni ataku i oceny dostępnych usług sieciowych.

Można również podjąć wewnętrzne testowanie sieci w celu zidentyfikowania i wykorzystania znanych podatności w warstwach 2 i 3 modelu OSI (takich jak zatruwanie buforu ARP czy 802.1Q *VLAN hopping*). W rozdziale 5 omówimy techniki odkrywania i oceny sieci, które można wykorzystać w środowiskach lokalnych do zidentyfikowania słabości.

Testowanie aplikacji Web

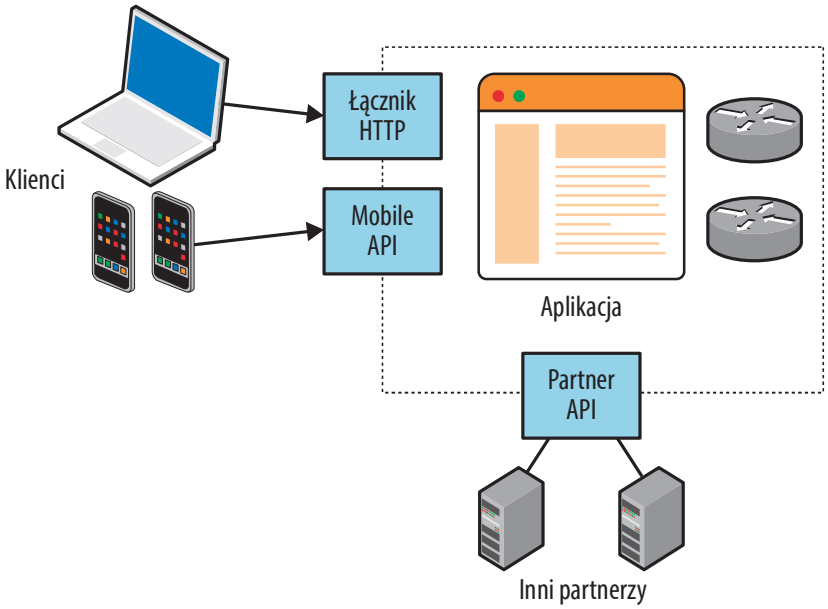
Dostępna zewnętrznie logika aplikacji Web jest oceniana albo w trybie z uwierzytelnieniem, albo bez niego. Większość organizacji wybiera testowanie uwierzytelnione, emulując napastnika dysponującego poprawnymi poświadczeniami lub tokenem sesji i szukającego możliwości eskalacji swoich uprawnień.

Publikowana przez Open Web Application Security Project (OWASP) lista Top 10 (*http://bit.ly/1lE9VSQ*) prezentuje typowe podatności aplikacji. Narzędzia, które potrafią wiarygodnie identyfikować i testować pod kątem tych podatności obejmują Burp Suite, IBM Security AppScan, HP WebInspect oraz Acunetix. Narzędzia te udostępniają funkcjonalność testową, która pozwala ogólnie przeskanować udostępnianą logikę aplikacji pod kątem problemów OWASP Top 10, w tym skryptowanie międzylokacyjne (cross-site scripting – XSS), międzylokacyjne fałszowanie żądań (crosssite request forgery – CSRF), wstrzykiwanie poleceń (iniekcje kodu SQL, poleceń powłoki systemu operacyjnego oraz LDAP), słabości zarządzania sesjami oraz błędy wycieku informacji. Pogłębione testowanie aplikacji Web wykracza poza tematykę tej książki, choć w rozdziałach 12 do 14 znajdziemy omówienie oceny serwerów Web i platform aplikacji.

W latach 2012-2013 pracowałem nad reakcją na incydenty i prowadziłem dochodzenia w firmach, które padły ofiarą ataku przeprowadzonego przez Alexseya Belana (*http://bit.ly/2alPf4O*). W każdym przypadku skompromitował on wewnętrzne aplikacje Web w celu podniesienia uprawnień i przechodzenia do równolegle działających komputerów. Podkreśla to ważność testów i wzmacniania aplikacji Web także w środowisku, które nie jest bezpośrednio dostępne z Internetu.

Testowanie usług Web

Aplikacje Web oraz mobilne wykorzystują API po stronie serwera, przy czym coraz większa część przetwarzania i renderowania jest wykonywana w systemie klienckim. API te są często udostępniane końcowym użytkownikom (jak w przypadku korzystania z mobilnego klienta bankowości online), innym firmom (partnerom biznesowym lub firmom powiązanym), a także innym wewnętrznym komponentom aplikacji, co pokazuje rysunek 1-2.



Rysunek 1-2 *Usługi Web wykorzystywane w aplikacji Web*

API typu REST są prezentowane w wielu aplikacjach za pośrednictwem protokołów takich jak XML-RPC, SOAP i inne. REST wykorzystuje zaawansowaną funkcjonalność HTTP (w tym funkcje buforowania i zachowania ciągłości), dzięki czemu projektowanie aplikacji i skalowanie implementacji jest prostsze.

Socjotechnika oparta o Internet

W czasie mojej kariery zawodowej niektóre z największych włamań, które przeprowadziłem podczas testów, opierały się na socjotechnice wykorzystującej Internet. Dwa typowe scenariusze ataku wyglądały następująco:

- Skonfigurowanie serwera Web udającego uprawniony zasób i wysłanie do wybranych użytkowników wiadomości zawierających łącze do podstępnej strony.
- Wysłanie spreparowanego materiału (np. dokumentu zawierającego kod exploitu dla Microsoft Excel lub Adobe Acrobat Reader) bezpośrednio do użytkownika za pośrednictwem wiadomości email, komunikatora lub inną metodą z pozornie zaufanego źródła, takiego jak przyjaciel lub kolega z pracy.

W roku 2012 wykonałem testy phishingu na zlecenie organizacji usług finansowych, wykorzystując fałszywy punkt końcowy SSL VPN i wysyłając do 200 użytkowników emaile z instrukcją zalogowania się do „nowej bramki VPN korporacji". W ciągu dwóch godzin 13 wpisało swoje nazwy użytkowników Active Directory, hasło domenowe oraz wartość tokenu drugiego składnika uwierzytelniającego. W rozdziale 9 zajmiemy się szczegółowo taktykami i narzędziami wykorzystywanymi w phishingu.

O czym jest ta książka

Książka obejmuje dynamiczne testowanie urządzeń sieciowych, systemów operacyjnych i wystawionych usług, natomiast pomija zagadnienia analiz statycznych i audytu. Testowanie aplikacji Web (czyli ocenę niestandardowych usług sieciowych i komponentów aplikacji) wykracza poza tematykę, podobnie jak VoIP lub oceny protokołów bezprzewodowych z rodziny 802.11. Te trzy zagadnienia już wypełniają całe książki, w tym:

- *The Web Application Hacker's Handbook*, Dafydd Stuttard i Marcus Pinto, Wiley 2011
- *Hacking Exposed Unified Communications & VoIP*, Mark Collier i David Endler, McGraw-Hill 2013
- *Hacking Exposed Wireless*, Johnny Cache, Joshua Wright i Vincent Liu, McGraw-Hill 2010

Rozdział 2

Przebieg oceny i narzędzia

Rozdział ten zawiera zarys podejścia polegającego na wyczerpujących testach penetracyjnych, wraz z przeglądem narzędzi obecnych w profesjonalnym arsenale. Wiele z tych programów można uruchomić tylko w systemach opartych na Unix, podczas gdy inne narzędzia specyficzne dla systemu Windows są niezbędne przy testowaniu technologii firmy Microsoft. Tym samym kluczowe jest zbudowanie elastycznej platformy testowej.

W firmie Matta[1] prowadziliśmy projekt o nazwie *Sentinel* dla dostawców usług testowania zabezpieczeń dla klientów z sektora usług finansowych. Platforma Sentinel zawierała pewną liczbę podatnych systemów, zaś dostawcy byli oceniani na podstawie tego, jakie podatności udało im się zidentyfikować. Wykonaliśmy oceny ponad 30 globalnych firm prowadzących testy penetracyjne i w pojedynczym teście obejmującym 10 testerów ustaliliśmy, że:

- Dwóch z nich nie wykonało skanowania wszystkich 65536 portów TCP.
- Pięciu nie zgłosiło faktu, że hasło roota usługi MySQL brzmiało „password".

Niektórzy testerzy byli oceniani więcej niż jeden raz. Dostrzegalny był brak stosowania ścisłej metodologii testów i wyniki ich badań (a konkretnie finalny raport przedstawiany klientowi) bardzo się zmieniały w zależności od zaangażowanych konsultantów.

W trakcie testowania ważne jest, aby pamiętać, że istnieje cała metodologia testów, której *należy* przestrzegać. Zbyt często inżynierowie i konsultanci wpadają w przysłowiową króliczą norę i pomijają kluczowe obszary analizowanego środowiska (takie jak niskopoziomowe testy interfejsów routera czy wystawione usługi pocztowe).

1 *https://www.trustmatta.com/*

Równie ważna jest też szybka identyfikacja najbardziej istotnych podatności środowiska. Z tego względu metodologia ta ma dwie cechy wyróżniające:

1. Wszechstronność pozwalająca na spójną identyfikację znaczących problemów.
2. Elastyczność pozwalająca na określenie priorytetów i zmaksymalizowanie uzyskiwanych wyników.

Metodologia oceny bezpieczeństwa sieci

Najlepsze praktyki metodologii oceny stosowane przez zdeterminowanych napastników i konsultantów zabezpieczeń obejmują cztery oddzielne kroki:

- Rozpoznanie w celu identyfikacji adresów IP sieci i hostów oraz użytkowników wchodzących w zakres naszych zainteresowań.
- Skanowanie podatności w celu wykrycia warunków, które potencjalnie można wykorzystać.
- Prześledzenie tych podatności i dalsze ręczne badanie sieci.
- Wykorzystanie podatności i przełamanie mechanizmów zabezpieczeń.

Metodologia ta jest odpowiednia dla połączonych z Internetem sieci testowanych na ślepo, przy ograniczonych informacjach o celu (na przykład tylko nazwa domeny). Jeśli konsultant został zatrudniony w celu oceny określonego bloku przestrzeni adresowej IP, może pominąć wstępne wyliczanie sieci i rozpocząć masowe skanowanie sieci i szukanie podatności.

Lokalne testowanie sieci obejmuje sprawdzenie nieroutowalnych protokołów oraz funkcjonalności 2 warstwy modelu OSI (np. kontrolę dostępu do sieci standardu 802.1X czy znakowanie VLAN standardu 802.1Q – *VLAN tagging*). Podczas testów z perspektywy lokalnego włamywacza wykorzystać można takie mechanizmy, jak *VLAN hopping*, podsłuchiwanie sieci oraz MITM. Lokalne poznawanie sieci i rodzaje ataków omówione są szczegółowo w rozdziale 5.

Rozpoznanie

Do identyfikowania interesujących nas hostów, sieci i użytkowników wykorzystywanych jest wiele taktyk. Ogólnie dostępne źródła obejmują wyszukiwarki internetowe, bazy danych WHOIS oraz serwery DNS. Poprzez odpytanie tych źródeł napastnik uzyskuje informacje o strukturze środowiska stanowiącego cel bez bezpośredniej interakcji z tym środowiskiem, takiej jak skanowanie portów.

Wstępny rekonesans może ujawnić hosty, które nie są odpowiednio ufortyfikowane. Zdeterminowani napastnicy inwestują swój czas w identyfikowanie sieci peryferyjnych i zawartych w nich hostów. Organizacje często koncentrują swoje wysiłki na zabezpieczaniu

oczywistych systemów publicznych (takich jak publiczny serwer Web lub poczty), pomijając hosty towarzyszące i wspomagające. W rezultacie te zlekceważone hosty mogą być prawdziwym prezentem dla napastników.

Użyteczne elementy informacji gromadzone we wstępnym rozpoznaniu obejmują szczegóły internetowych bloków sieciowych i wewnętrznych adresów IP. Poprzez analizę danych DNS i WHOIS można zacząć mapować sieci badanej organizacji i zrozumieć zależności pomiędzy fizycznymi lokalizacjami.

Informacje te są następnie wykorzystywane w fazach skanowania podatności i testów penetracyjnych, używanych w celu zidentyfikowania dających się wykorzystać luk w wystawionych komponentach. Dalsze rozpoznanie obejmuje wydobycie szczegółów użytkowników (takich jak adresy email, numery telefonów i nazwy użytkowników), którymi można się posłużyć w próbach siłowego złamania haseł i procesach socjotechnicznych.

Skanowanie podatności

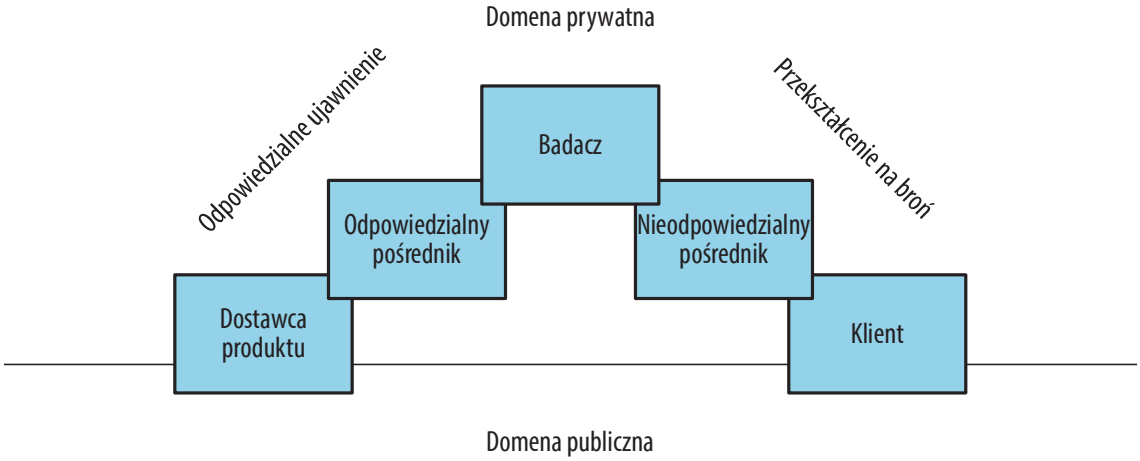
Po zidentyfikowaniu interesujących go bloków adresów IP napastnik przeprowadza masowe skanowanie w celu wykrycia dostępnych usług sieciowych, które można będzie później wykorzystać do osiągnięcia celów – może to być wykonanie własnego kodu, wyciek informacji, odmowa usługi lub uprzywilejowany dostęp do wybranych komponentów. Narzędzia skanowania sieci (takie jak Nmap, Nessus, Rapid7 Nexpose i QualysGuard) wykonują rozpoznawanie usług, badanie i testowanie pod kątem znanych problemów.

Użyteczne informacje gromadzone w fazie skanowania obejmują szczegóły dostępnych hostów i wystawionych usług sieciowych, uzupełnione o informacje dodatkowe (takie jak szczegóły komunikatów ICMP, na które odpowiadają hosty stanowiące cel, a także wgląd w listy kontroli dostępu zapór ogniowych). Narzędzia skanowania rozpoznają również znane słabości mogące potencjalnie występować wewnątrz dostępnych usług, które można i należy dokładniej zbadać.

Badanie podatności

Informacje o wykrytych słabych punktach oprogramowania są niekiedy ujawniane publicznie w jakimś forum internetowym lub na liście dyskusyjnej, ale coraz częściej są one sprzedawane takim organizacjom, jak Zero Day Initiative (ZDI), które z kolei odpowiedzialnie ujawniają problem dostawcy oprogramowania i powiadamiają swoich (płacących) subskrybentów. Zgodnie z danymi Immunity Inc. przeciętnie dowolny bug ma czas życia 348 dni, zanim będzie dostępna łatka przygotowana przez dostawcę.

Wiele prywatnych organizacji badawczych i pośredników nie powiadamia dostawców oprogramowania o wykrytych lukach, a niektóre udostępniają komercyjnie wspierane exploity *zero-day* swoim klientom. Pomiędzy odpowiedzialnym ujawnieniem, pierwszym użyciem i publicznym omówieniem znanych podatności szczegóły dotyczące luki są pofragmentowane, co pokazuje rysunek 2-1.



Rysunek 2-1 *Rozpowszechnianie się informacji o podatności w domenie publicznej*

W celu skutecznego wykrycia podatności w badanym środowisku profesjonalista zabezpieczeń powinien wiedzieć o istniejących bugach – zarówno tych dostępnych publicznie, jak i znanych tylko prywatnym organizacjom. Jednak wielu z nas nie ma dostępu do prywatnych kanałów informacji i opiera swoje oceny na wiedzy publicznej, uzupełnionej swoimi własnymi badaniami.

Publiczne źródła informacji o podatnościach

Podczas wyszukiwania potencjalnych luk przydatne są poniższe otwarte źródła informacji:

- NIST National Vulnerability Database (*http://nvd.nist.gov*)
- Open Sourced Vulnerability Database (*http://www.osvdb.org*)
- Offensive-Security Exploit Database (*http://www.exploit-db.com*)
- Lista mailowa Full Disclosure (*http://seclists.org/fulldisclosure/*)
- HackerOne, internetowa giełda bugów (*https://hackerone.com/internet*)
- SecurityFocus (*http://www.securityfocus.com*)
- Packet Storm (*http://packetstormsecurity.com*)
- CERT Vulnerability Notes Database (*http://www.kb.cert.org/vuls*)

ZDI i Project Zero firmy Google utrzymują publicznie dostępne bugtrackery, które szczegółowo opisują pojawiające się odkrycia i niezałatane podatności[2]. Otwarte projekty, takie jak OpenSSL oraz jądro Linuxa, również mają publiczne bugtrackery, które ujawniają przydatne szczegóły o niezałatanych słabościach. W czasie testów warto przeglądać zarówno bugtracki, jak i uwagi do wydania, aby poznać słabości obecne w pakietach oprogramowania.

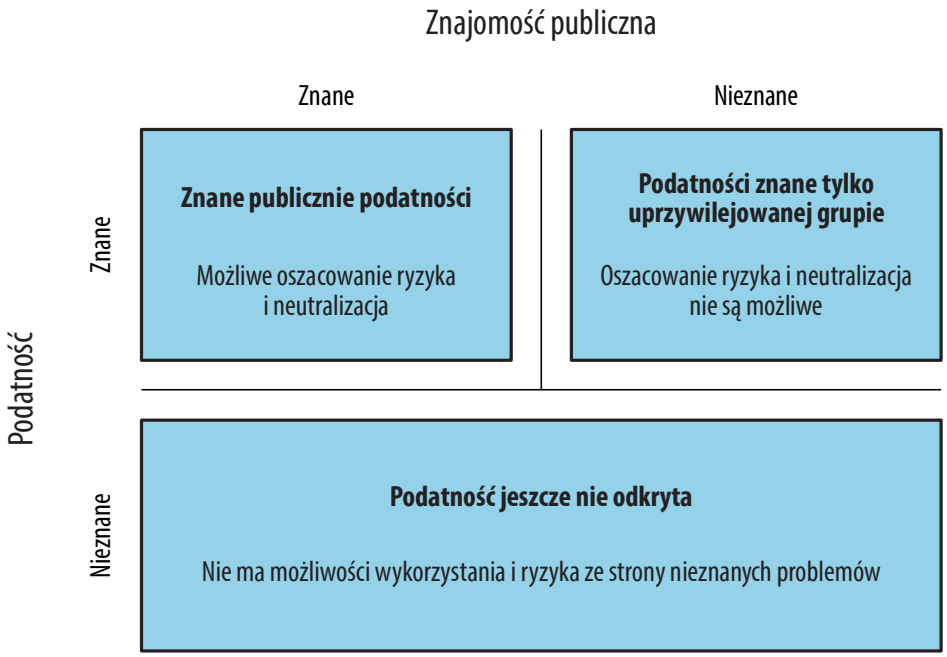
2 Lista zaleceń Zero Day Initiative (*http://bit.ly/2aA844y*) oraz strona „Chromium bugs" (*http://bit.ly/2aA7PGM*).

Narzędzia opracowywane przez społeczność, takie jak Rapid7 Nexpose, Nessus, Metasploit lub Nmap również udostępniają szczegóły podatności identyfikowanych podczas testów. Posługując się tymi źródłami można uzyskać solidne pokrycie publicznie znanych podatności i zagrożeń.

Prywatne źródła informacji o podatnościach

Rozmaite organizacje, w tym agencje rządowe i instytucje przemysłu obronnego, uzyskują poufne informacje o podatnościach od brokerów i źródeł takich jak ZERODIUM, Exodus Intelligence, Netragard i ReVuln. Organizacje te są znane z tego, że prowadzą własne programy badawcze pod kątem podatności i luk w oprogramowaniu, udostępniając informacje o nienaprawionych błędach swoim subskrybentom.

Stefan Frei z NSS Labs opublikował ciekawy artykuł na ten temat[3]. W swoim opracowaniu Frei omawia podatności znane uprzywilejowanej grupie, co demonstruje rysunek 2-2.



Rysunek 2-2 *Macierz odkrywania i ujawniania podatności*

Opierając się na ujawnianych w mediach informacjach o budżecie rządu Stanów Zjednoczonych na zakup informacji o exploitach oraz na przeciętnych cenach takich informacji uzyskanych od brokerów Frei oszacował, że w dowolnym dniu istnieje co najmniej 85 „znanych nieznanych". Zależnie od tego, jakie strony są zaangażowane i jakie są zasady ich funkcjonowania, bugi te mogą nigdy nie zostać ujawnione producentom oprogramowania.

3 Stefan Frei, „The Known Unknowns & Outbidding Cybercriminals", Swiss Federal Institute of Technology Zurich, wrzesień 2014 (*http://bit.ly/2aNEYSG*)

Analiza przypadku rynku exploitów *zero-day* opracowana przez Vlada Tsyrklevicha[4] daje wgląd w prywatny rynek informacji o podatności, w tym indywidualne ceny exploitów oraz szczegóły nienaprawionych luk w takich pakietach oprogramowania, jak Adobe Flash, Microsoft Office 2007, Internet Explorer 11 czy Oracle Database.

Wykorzystanie podatności

Po zidentyfikowaniu potencjalnej podatności napastnik może wykorzystać słabość wystawionego kodu w celu osiągnięcia swoich celów. Zależnie od konkretnych celów, może też wykorzystać podatności do uzyskania dostępu do zabezpieczonej sieci prywatnej lub zdobycia poufnych informacji.

W trakcie testu penetracyjnego zakwalifikowanie podatności zazwyczaj obejmuje jej wykorzystanie. Istnieją solidne, komercyjnie wspierane platformy zapewniające elastyczne wybieranie podatnych komponentów w ramach badanego środowiska (obsługujące różne systemy operacyjne i konfiguracje), które pozwalają na użycie konkretnych exploitów. Dodatkowo używane są opracowane przez niezależnych dostawców moduły rozszerzające te platformy, zapewniające wsparcie dla SCADA[5] i innych technologii. Do popularnych platform wykorzystujących exploity należą:

- Rapid7 Metasploit (*http://www.rapid7.com/products/metasploit/*)
- CORE Impact (*http://www.coresecurity.com/core-impact-pro*)
- Immunity CANVAS (*https://www.immunitysec.com/products-canvas.shtml*)

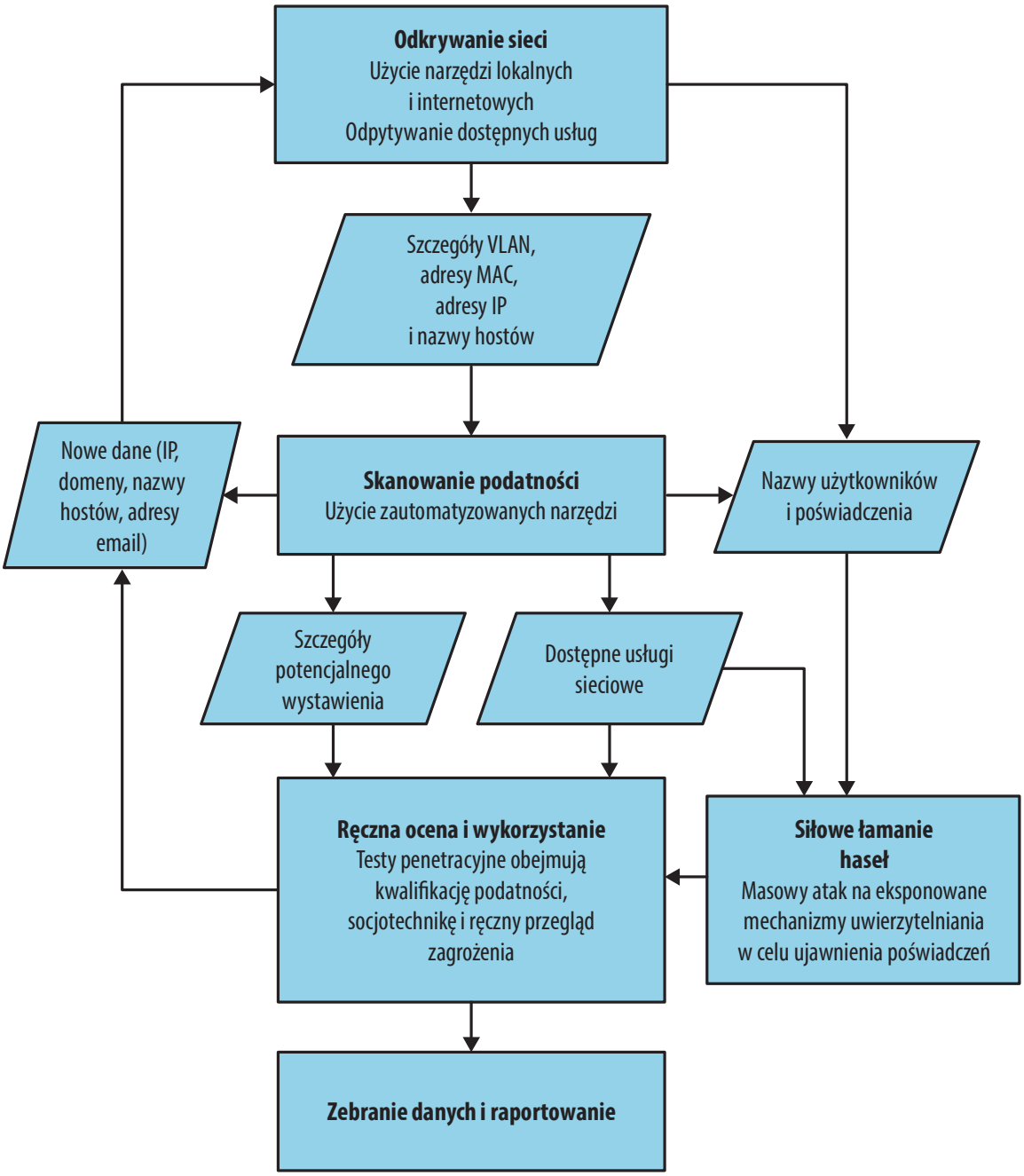
Zgodnie z dokumentem NIST SP 800-115 zadania wykorzystania podatności należą do kategorii *Target Vulnerability Validation Techniques* (techniki weryfikacji podatności celu). Jako tester (posiadający autoryzację organizacji stanowiącej cel) możemy podejmować próby łamania haseł, socjotechniki i kwalifikowania podatności za pomocą tych platform i narzędzi.

Iteracyjne podejście do oceny

Proces oceny jest z natury iteracyjny, przy czym nowe dane (np. bloki adresów IP, nazwy hostów DNS lub poświadczenia użytkowników) są często wprowadzane ponownie do wcześniejszych faz przebiegu pracy. Rysunek 2-3 prezentuje szkic tego podejścia, pokazując dane przekazywane pomiędzy poszczególnymi elementami testów.

4 *http://bit.ly/2aNFro6*

5 *http://scadavulns.com/*



Rysunek 2-3 *Podejście iteracyjne do odkrywania i testów*

Platforma testowa

Aby móc zapewnić korzystanie z narzędzi testowych specyficznych dla różnych systemów operacyjnych, najlepszym wyjściem jest posłużenie się oprogramowaniem wirtualizacyjnym, takim jak VMware Fusion dla Mac OS X czy VMware Workstation dla Windows. Użyteczne natywne aplikacje dostępne w systemach Linux obejmują *showmount*, *dig* oraz *snmpwalk*. Możliwe jest też budowanie narzędzi skanujących i testujących specyficzne, dostępne w sieci usługi oraz aplikacje Web.

Kali Linux jest dystrybucją zorientowaną na testy penetracyjne, którą można łatwo uruchomić w wirtualnym środowisku. Kali zawiera większość narzędzi przedstawianych w tej książce, w tym Metasploit, Nmap, Burp Suite oraz Nikto. Dokumentacja Kali Linux[6] zawiera omówienie instalacji, a ponadto niedawno opublikowane zostały dwie książki szczegółowo opisujące działanie i stosowanie indywidualnych narzędzi:

- *Kali Linux: Assuring Security by Penetration Testing*, Tedi Heriyanto i inni, Packt Publishing, 2014.
- *Mastering Kali Linux for Advanced Penetration Testing*, Robert W. Beggs, Packt Publishing, 2014

Jeśli na przykład posługujemy się komputerem systemu Mac OS X, uruchomienie Microsoft Windows 7 oraz Kali Linux jako maszyn wirtualnych VMware Fusion będzie wystarczające w większości zastosowań. Firma Offensive Security utrzymuje niestandardowe obrazy Kali Linux dla VMware i VirtualBox[7], obejmujące natywne wsparcie dla VMware Tools (pozwalające na wykonywanie kopiowania i wklejania pomiędzy wirtualnym systemem gościa a systemem hosta), oraz urządzeń z procesorami ARM, takimi jak Chromebook.

Aktualizowanie Kali Linux

Po zainstalowaniu Kali Linux konieczne jest wykonanie aktualizacji dystrybucji i pakietów. W szczególności nowe wydania Nmap oraz Metasploit zawierają komponenty, które powinniśmy aktywnie kontrolować. Aktualizację systemu umożliwiają poniższe polecenia:

```
apt-get update
apt-get dist-upgrade
updatedb
```

6 *http://docs.kali.org/*

7 *https://www.offensive-security.com/kali-linux-vmware-arm-image-download/*

Wdrażanie podatnego serwera

W celu poznania i przetestowania narzędzi w kontrolowanym środowisku można wykorzystać obraz podatnego serwera, względem którego będzie można przeprowadzić skanowanie i upewnić się, że nasz przybornik jest właściwie skonfigurowany i działa poprawnie. Firma Rapid7 przygotowała obraz maszyny wirtualnej o nazwie *Metasploitable 2*[8], która poddaje się atakom z wielu stron, w tym:

- Tylne furtki w pakietach, w tym FTP oraz IRC.
- Podatne usługi Unix RPC.
- Eskalacja uprawnień SMB.
- Słabe hasła użytkowników.
- Problemy w konfiguracji serwera Web i aplikacji (poprzez Apache HTTP Server oraz Tomcat).
- Podatności aplikacji Web (np. phpMyAdmin oraz TWiki).

Istnieją liczne tutoriale i przewodniki online opisujące narzędzia, moduły Metasploit oraz polecenia, które można wykorzystać do sprawdzenia i wykorzystania podatności obecnych w maszynie wirtualnej. Dwa szczególnie użyteczne zasoby sieciowe to:

- *Pentest Lab – Metasploitable-2* (*http://bit.ly/2aNFbFn*).
- *Computer Security Student* (należy przejść do Security Tools→Metasploitable Project→Exploits) (*http://www.computersecuritystudent.com/*)

Istnieją też inne obrazy podatnych maszyn wirtualnych, które można znaleźć w witrynie VulnHub[9]. W przypadku testowania aplikacji Web warto poświęcić nieco czasu na przejrzenie katalogu podatnych aplikacji udostępnianego przez OWASP[10], a w szczególności na ćwiczenia oferowane przez *PentesterLab*[11].

8 Więcej informacji zawiera „Metasploitable 2 Exploitability Guide" (*http://bit.ly/2aNF8cY*).
9 *https://www.vulnhub.com*
10 *http://bit.ly/2aNEMD9*
11 *https://pentesterlab.com/exercises/*

Rozdział 3

Luki i przeciwnicy

> Ilekroć próbujesz zrobić porządne przestępstwo, istnieje pięćdziesiąt sposobów, aby wszystko spieprzyć. Gdybyś zdołał przewidzieć dwadzieścia pięć z nich, byłbyś geniuszem. A nie jesteś.
>
> *– Mickey Rourke*
> *Żar ciała (1981)*

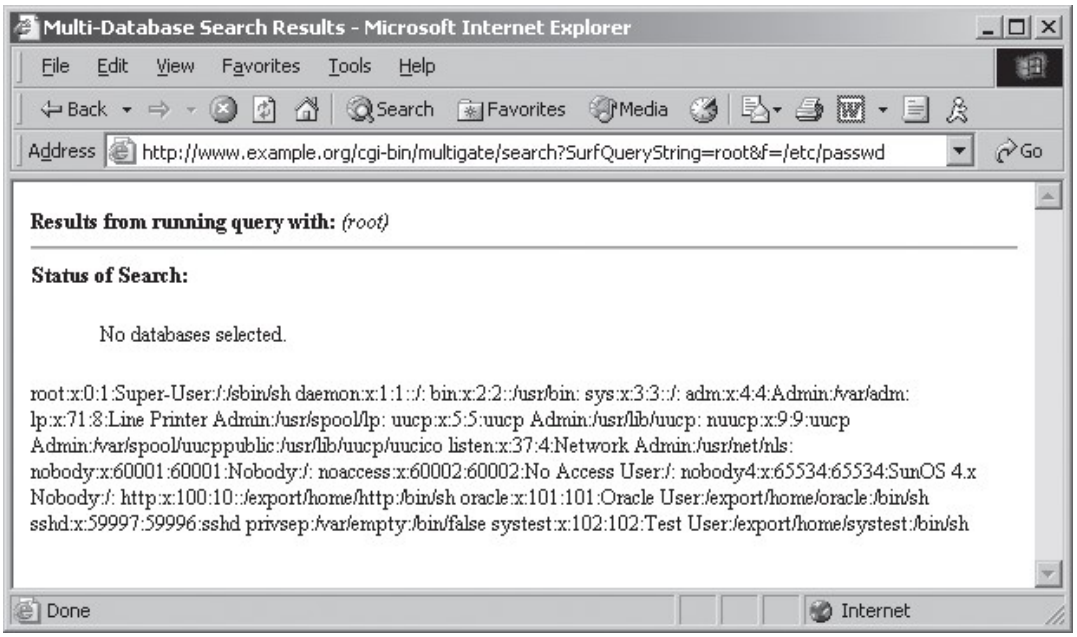
Inżynierowie oprogramowania budują coraz bardziej skomplikowane systemy bez rozważania nawet połowy sposobów, by *coś się spieprzyło*. Dzięki przetwarzaniu w chmurze i rosnącej liczby warstw abstrakcji nie dość zabezpieczone produkty nadal będą istnieć, a biznes zabezpieczania danych pozostanie lukratywny w kolejnych latach.

Fundamentalne koncepcje hackingu

Hacking to sztuka manipulowania systemem w celu wykonania niezaplanowanego zadania.

Prostym przykładem może być mechanizm wyszukiwania: program przekazuje zapytanie dostarczone przez użytkownika do bazy danych i zwraca listę wyników. Całe przetwarzanie następuje po stronie serwera. Dzięki zrozumieniu, jak projektowane są wyszukiwarki i znajomości ich potencjalnych pułapek (takie jak przyjmowanie zarówno łańcucha zapytania, jak i wartości źródłowej do tego zapytania) przeciwnik może próbować zmanipulować aplikację i uzyskać dostęp do poufnych plików.

Kilka lat temu witryny Pentagonu, Air Force oraz Navy miały właśnie taki problem – wykorzystanie mechanizmu wyszukiwania o nazwie *multigate*, akceptującego dwa argumenty: `SurfQueryString` oraz `f`. Dzięki odpowiednio skonstruowanemu adresowi URL została ujawniona zawartość lokalnego pliku */etc/passwd*, co pokazuje rysunek 3-1.



Rysunek 3-1 *Manipulowanie mechanizmem wyszukiwania multigate*

Witryny te były chronione w warstwie sieci przez zapory ogniowe i specjalizowany sprzęt zabezpieczający. Jednak ze względu wielką ilość przechowywanych informacji zaimplementowano w nich mechanizm wyszukiwania, co stworzyło lukę w warstwie aplikacji.

Użyteczną metodą analizy procesu wykorzystania eksploita jest zastanowienie się, jak napastnik mógłby chcieć przeprogramować system, aby wykonać użyteczne działanie. Przyjmowane taktyki będą zależne od elementów składowych systemu, dostępnych funkcjonalności i celów napastnika – takich jak ujawnienie danych, podniesienie uprawnień, wykonanie wybranego kodu lub wyłączenie usługi.

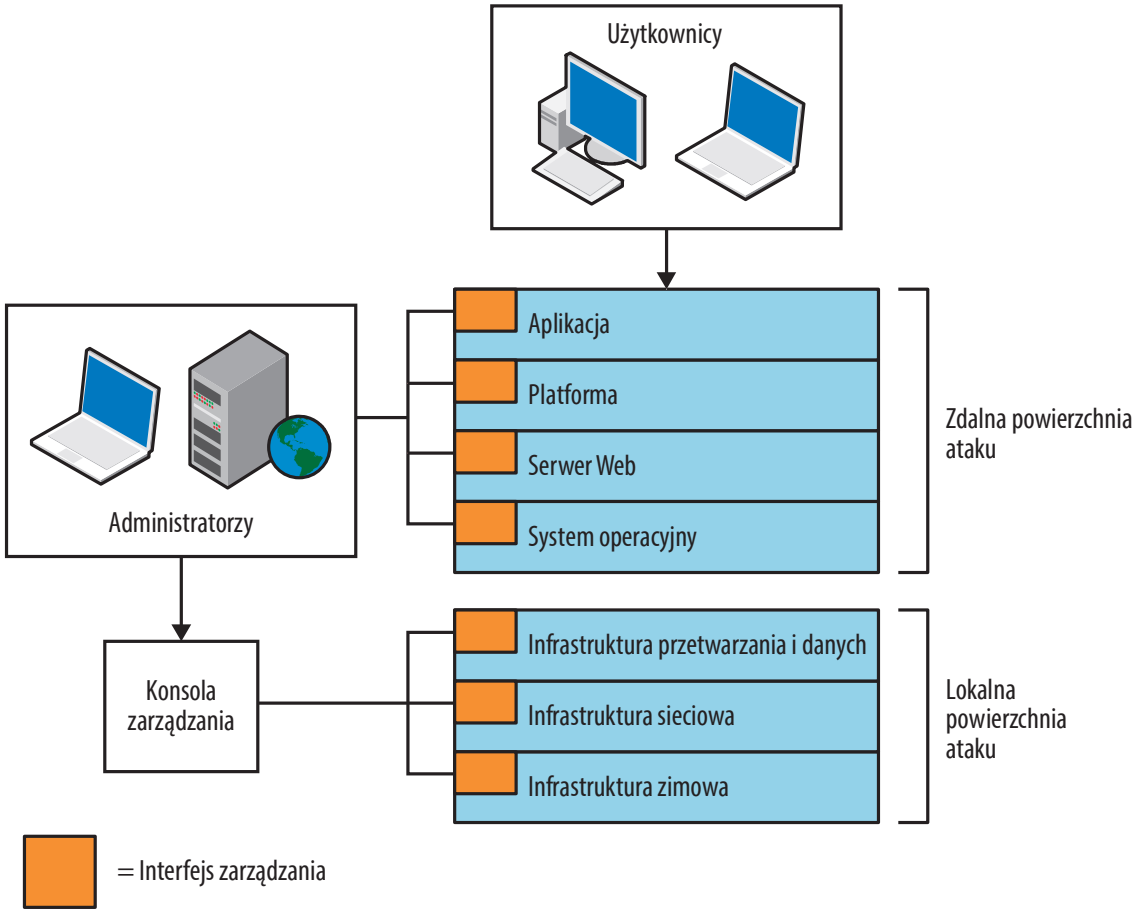
Dlaczego oprogramowanie jest podatne

Od 30 lat napastnicy wykorzystują odkrywane słabości oprogramowania, zaś producenci pracują nad poprawieniem bezpieczeństwa ich produktów i systemów operacyjnych. W coraz bardziej złożonych systemach zawsze będą istnieć jakieś usterki i wady – szczególnie że gwarancje jakości są niewystarczające.

Producenci oprogramowania znaleźli się w sytuacji, że w miarę wzrostu złożoności ich produktów zabezpieczanie ich przed spodziewanym atakiem staje się kosztowne. Statyczny przegląd kodu oraz dynamiczne testy oprogramowania są znaczącymi przedsięwzięciami, które mają poważny wpływ na podstawowe interesy firmy i opóźniają dostarczenie produktu na rynek.

Rozważanie powierzchni ataku

W każdym udanym ataku na system komputerowy napastnik do osiągnięcia swojego celu wykorzystał dostępną powierzchnię ataku. Powierzchnia ta często obejmuje aplikacje serwerowe, klienckie punkty końcowe, samych użytkowników, kanały komunikacyjne oraz elementy infrastruktury. Dowolny udostępniony komponent budujący większy system może być celem.



Rysunek 3-2 *Powierzchnia ataku aplikacji chmurowej*

Praktyczne sposoby umożliwiające infiltrację środowiska obejmują:

- Skompromitowanie dostawcy hostingu lub konsoli zarządzania chmurą[1].
- Luka wewnątrz infrastruktury (np. atak poprzez kanał hyperwizora[2]).

1 John Leyden, „Linode Hackers Escape with $70k in Daring Bitcoin Heist", The Register, 2 marca 2012 (*http://bit.ly/2aNGNPn*).

2 Yinqian Zhang i in., „Cross-VM Side Channels and Their Use to Extract Private Keys", wykład przedstawiony na *2012 ACM Conference on Computer and Communications Security* w Raleigh, North Carolina, 16–18 października 2012 (*http://unc.live/2aNGqVd*).

- Podatność systemu operacyjnego (taka jak wada sterownika sieciowego lub błąd kernela).
- Wada oprogramowania serwerowego (np. bugi bibliotek Nginx lub OpenSSL).
- Wada platformy aplikacji Web (np. defekt zarządzania sesjami).
- Bug aplikacji Web (np. iniekcja kodu lub błąd przebiegu logiki biznesowej).
- Atak na aktywne sesje użytkowników poprzez HTTPS lub SSH.
- Przechwycenie poprawnych poświadczeń (np. klucza SSH lub tokenu uwierzytelniającego).
- Atak na oprogramowanie klienckie (np. klienta SSH PuTTY lub wykorzystanie luki w przeglądarce).
- Atak na użytkownika poprzez socjotechnikę lub przechwycenie kliknięć.

Niektóre taktyki ataku są stosowane zdalnie, podczas gdy inne mogą wynikać z lokalnego sąsiedztwa lub dostępu do sieci, aby możliwa była interakcja z określonymi komponentami systemu. Na przykład w środowiskach chmurowych o wielu dzierżawcach uzyskanie dostępu do wewnętrznej przestrzeni adresowej często ujawnia interfejsy, które nie są dostępne publicznie.

Taksonomia błędów zabezpieczeń oprogramowania

Rysunek 3-2 demonstruje, jak indywidualne składniki oprogramowania mogą utworzyć wielką powierzchnię ataku. Lokalnie wewnątrz każdego elementu programowego mogą istnieć błędy zabezpieczeń, które mogą zostać nadużyte. W 2005 roku Gary McGraw, Katrina Tsipenyuk oraz Brian Chess opublikowali taksonomię[3], której można użyć do kategoryzowania niedostatków oprogramowania. Taksonomia ta obejmuje siedem królestw:

Weryfikacja i prezentowanie danych wejściowych
: Niezweryfikowanie danych wejściowych lub niewłaściwe ich kodowanie, co prowadzi do przetworzenia złośliwej zawartości. Przykłady gromad gatunków wewnątrz tego królestwa obejmują przepełnienia bufora, skryptowanie międzylokacyjne (XSS), iniekcje kodu oraz zewnętrzne przetwarzanie jednostek XML (XXE).

Nadużycie API
: Napastnicy wykorzystują funkcje wystawiane przez biblioteki lokalnego systemu i udostępnione API do wykonania arbitralnego kodu, odczytywania poufnych danych lub pominięcia mechanizmów zabezpieczeń.

3 Katrina Tsipenyuk, Brian Chess i Gary McGraw, „Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors", *IEEE Security & Privacy* listopad/grudzień 2005, (*http://bit.ly/2aNGX9H*)

Funkcje zabezpieczeń
: Projekt i implementacja funkcji zabezpieczeń wewnątrz oprogramowania powinny być solidne. Niezapewnienie właściwego generowania liczb losowych, wymuszenia najmniejszych przywilejów czy bezpiecznego przechowywania wrażliwych danych często prowadzi do kompromitacji.

Czas i stan
: Podatności dotyczące czasu i stanu aplikacji mogą zostać nadużyte poprzez stworzenie warunków wyścigu, dzięki istnieniu niezabezpieczonych plików tymczasowych lub poprzez utrwalenie sesji w aplikacji Web (czyli nie przedstawienie nowego tokenu sesji podczas logowania).

Błędy
: Po znalezieniu sytuacji powodującej błąd napastnik może zaatakować kod obsługi błędu w celu wpłynięcia na logiczny przebieg programu lub odczytania poufnych danych.

Jakość kodu
: Niska jakość kodu prowadzi do nieprzewidywalnego zachowania w aplikacji. W przypadku języków niskiego poziomu (takich jak C/C++) tego typu zachowania mogą być katastrofalne, prowadząc do wycieków pamięci i wykonania dowolnego kodu.

Kapsułkowanie
: Dwuznaczność metod, jakimi aplikacja zapewnia dostęp zarówno do ścieżek danych, jak i wykonania kodu, może prowadzić do luk kapsułkowania. Przykłady obejmują wycieki danych pomiędzy użytkownikami lub pozostawionego w aplikacji kodu debugowania, który może zostać wykorzystany przez napastnika.

Ósmym królestwem jest *środowisko* – tym terminem klasyfikowane są podatności występujące poza kodem źródłowym oprogramowania (np. błędy w oprogramowaniu interpretera lub kompilatora, platformie aplikacji Web lub w infrastrukturze sprzętowej).

Taksonomii tej należy używać do definiowania źródeł defektów, które prowadzą do nieoczekiwanego zachowania aplikacji. Królestwa te nie są stosowane do definiowania klas ataków (takie jak ataki przez boczny kanał).

Modelowanie zagrożeń

Napastnicy wyszukują i eksploatują słabości wewnątrz różnych składników i funkcji systemu. Taksonomia słabości pozwala opisać i skategoryzować niskopoziomowe błędy w każdym pakiecie oprogramowania, ale nie obejmują szerszych problemów środowiskowych