Adam KOZAKIEWICZ[1]

FAŁSZYWE ALARMY W WYSOKOINTERAKTYWNYCH KLIENCKICH HONEYPOTACH

Streszczenie

Klienckie honeypoty to podstawowe narzędzie do wykrywania zagrożeń dla aplikacji klienckich. Artykuł prezentuje wprowadzenie do tematyki, koncentrując się na wysokointeraktywnych klienckich honeypotach, wykorzystujących kompletne systemy w roli pułapek. Przedstawiony został problem występowania fałszywych alarmów i jego konsekwencje dla użytkowników honeypotów. Artykuł omawia oryginalne podejście do eliminacji tego problemu, oparte na metodach sztucznej inteligencji, przedstawiając zarys sposobu działania i wyniki najważniejszych testów.

Abstract

Client honeypots are a key tool for detection of threats to client applications. This paper presents a short introduction to this area of research, concentrating on high-interaction client honeypots, which use complete systems as traps. The problem of false positives is stated, along with a description of its consequences for honeypot users. The paper demonstrates an original approach to dealing with this problem, based on artificial intelligence techniques. A brief overview of the method and results of the most important tests are shown.

1 WPROWADZENIE

Współczesny użytkownik komputera bardzo rzadko ogranicza się do korzystania z lokalnych zasobów maszyny. Szybki rozwój Internetu sprawił, że obecnie korzystanie z jego zasobów jest podstawowym zastosowaniem komputerów biurkowych. Ogólny postęp technologiczny sprawił też, że coraz więcej ludzi ma na co dzień dostęp do własnego komputera. Taki stan rzeczy istotnie wpływa na krajobraz zagrożeń dla bezpieczeństwa komputerów.

[1] Dr inż. Adam Kozakiewicz jest kierownikiem Zespołu Metod Bezpieczeństwa Sieci i Informacji w Pionie Naukowym Naukowej i Akademickiej Sieci Komputerowej – Instytutu Badawczego. Wykłada też w Warszawskiej Wyższej Szkole Informatyki i na Wydziale Elektroniki i Technik Informacyjnych Instytutu Automatyki i Informatyki Stosowanej.

Jeszcze w latach 90-tych głównym zagrożeniem były wirusy propagujące się przez dyskietki. W wyniku rozwoju sieci już na przełomie wieków zostały one wyparte przez zagrożenia korzystające w celu propagacji z Internetu. W tamtych czasach jednak przeciętny użytkownik korzystał z sieci w dość ograniczonym zakresie – należało zatem wykorzystywać przede wszystkim sam fakt posiadania przez większość komputerów połączenia z Internetem. Szybki rozwój przeżywały robaki internetowe, na oślep poszukujące w sieci podatnych aplikacji serwerowych; znaczącą rolę odgrywały też wirusy pocztowe. Jeszcze w 2004 roku za największe zagrożenie nadchodzących lat uważano dalszy rozwój tego typu ataków. Świat bezpieczeństwa skupił się więc na wykrywaniu i blokowaniu niechcianych połączeń i spamu. Tymczasem jednak autorzy złośliwego oprogramowania zauważyli zmiany zachodzące w społeczności internetowej i całkowicie zmienili wektor ataku. Liczba zwykłych biurkowych komputerów stała się tak olbrzymia, że stanowiły one łakomy kąsek, a przeciętny użytkownik coraz więcej czasu spędzał przeglądając zasoby Internetu przy użyciu przeglądarki – może szybkiej i wygodnej, ale pełnej błędów i korzystającej z mnóstwa, również pełnych błędów wtyczek. Po co w tej sytuacji ryzykować detekcję prowadzonych na oślep skanowań, skoro wystarczy w miarę dobrze się ukryć i czekać na wizytę ofiary.

Obecnie to właśnie ataki na luki w aplikacjach klienckich stanowią największe i najczęściej spotykane zagrożenie. W szczególności zagrożone są zaś przeglądarki internetowe. Klasyczne honeypoty serwerowe nie były w stanie wykryć i zarejestrować tego typu ataków. Opracowano więc nową grupę narzędzi: klienckie honeypoty.

Tematem niniejszego artykułu są zasady działania podgrupy tych narzędzi – wysokointeraktywnych klienckich honeypotów. W szczególności skupimy się na problemie ograniczania liczby fałszywych alarmów, prezentując opracowane w NASK rozwiązanie tego problemu.

Rozdział 2 przedstawia podział klienckich honeypotów na kategorie. Dla jednej z nich – honeypotów wysokointeraktywnych – w rozdziale 3 przedstawiono przegląd sposobów realizacji detekcji złośliwego oprogramowania. Rozdział 4 przedstawia bardziej szczegółowo interesujące nas rozwiązanie – Capture-HPC. Występujący w nim problem fałszywych alarmów jest omówiony w rozdziale 5, w którym zaprezentowano też ogólne zasady działania rozwiązania opracowanego w NASK. Artykuł kończy się krótkim podsumowaniem w rozdziale 6.

2 KLASYFIKACJA KLIENCKICH HONEYPOTÓW

Wyróżniamy dwie podstawowe grupy klienckich honeypotów – nisko- i wysokointeraktywne. Spotykane są również rozwiązania hybrydowe, łączące cechy obu grup.

Kryterium podziału jest sposób wchodzenia w interakcję z atakującym serwerem, czyli dokładność odwzorowania zachowania zwykłego komputera użytkownika Internetu.

Niskointeraktywne klienckie honeypoty stanowią w istocie prostą adaptację istniejących narzędzi do nowego celu. Zadanie odwiedzania podejrzanych stron powierzane jest crawlerom, czyli programom zaprojektowanym do automatycznego pobierania zawartości stron (były też nieliczne próby zaadaptowania do tego celu serwerów proxy i monitorowania przy ich użyciu ruchu generowanego przez niezmodyfikowane crawlery, lub wręcz zwykłych użytkowników). Crawler nie jest oczywiście przeglądarką, toteż autorzy malware'u mają szansę wykrycia, że odwiedzający nie jest potencjalną ofiarą, co obniża szansę wykrycia zagrożeń. Autorzy niskointeraktywnych klienckich honeypotów poświęcają zatem wiele pracy skutecznemu upodobnieniu się do przeglądarki. W skrajnych przypadkach crawler bywa wręcz integrowany z jedną z dostępnych na rynku przeglądarek, a przynajmniej z jej silnikiem javascriptowym. Rozwiązania takie można potraktować, jako pośrednie między nisko- a wysokointeraktywnymi – w skrajnych przypadkach mogą nawet same być podatne na niektóre z ataków, które miały wykrywać. Do detekcji zagrożeń bywają stosowane różne metody, od silników antywirusowych, poprzez systemy IDS, aż do bardzo złożonych heurystyk. Dodatkowa funkcjonalność obejmuje różne heurystyczne metody dokładnej identyfikacji wykrytych zagrożeń.

Zaletą niskointeraktywnych klienckich honeypotów jest przede wszystkim wydajność. Teoretycznie detekcja ograniczona jest do znanych zagrożeń, choć dobre heurystyki mogą wykryć niekiedy także ataki wcześniej nieznane. Wadą natomiast jest niezdolność (w większości wypadków) dotarcia do właściwego malware'u w związku z niemożnością wykonania shellcode'u wysłanego przez stronę, lub nie przesłaniem go w wyniku wykrycia próby detekcji. Wyraźny jest też trudny kompromis między możliwościami skutecznej detekcji a liczbą fałszywych alarmów – rezygnacja z heurystyk silnie ogranicza skuteczność, ale heurystyki z definicji mogą się mylić, a wyniki w przypadku błędu są trudne do automatycznej weryfikacji.

Wysokointeraktywne klienckie honeypoty rozwiązują problem skutecznego naśladowania rzeczywistej przeglądarki poprzez wykorzystanie oryginalnego, choć automatycznie sterowanego programu, działającego pod kontrolą rzeczywistego systemu operacyjnego. Detekcja sprowadza się zatem do wystawienia na zagrożenie rzeczywistej, najczęściej wirtualnej maszyny, która niczym z pozoru nie różni się od zwykłego komputera, a następnie obserwacji skutków tej ekspozycji. Ponieważ przez takie działanie dopuszcza się do rzeczywistej infekcji, konieczne jest czyszczenie systemu honeypota po każdej wykrytej infekcji, a na wszelki wypadek okresowo także w przypadku jej niewykrycia. Skutkiem tego rozwiązania jest ograniczona wydajność, wymuszająca korzystanie jednocześnie z wielu maszyn, co znacznie podnosi koszty. Teoretycznie w zamian otrzymuje

się niemal pewność detekcji – to oczywiście teoria, do infekcji nie dojdzie przecież, jeśli wykorzystywana przez daną stronę luka nie istnieje w wersji systemu lub przeglądarki zainstalowanych w honeypocie. Możliwy do uzyskania jest też spadek liczby fałszywych alarmów – jest on uzależniony od zastosowanej metody detekcji, a ściślej od tego, na ile jednoznacznie jest ona w stanie stwierdzić, czy zachowanie maszyny przestało mieścić się w normie. W dalszej części artykułu skupiać się będziemy właśnie na tej grupie honeypotów.

Jak widać, oba rozwiązania w pewnej mierze się uzupełniają. Z tego powodu niektóre rozwiązania klienckich honeypotów łączą oba typy w jeden system. Są to honeypoty **hybrydowe,** a ich przykładem jest system Honey-Spider Network, o którym nieco więcej informacji można znaleźć w jednym z dalszych rozdziałów.

Na koniec tej klasyfikacji warto wspomnieć, że skanowanie nie musi się zakończyć sukcesem nawet, jeśli honeypot wysokointeraktywny jest podatny na występujące w sieci zagrożenie, albo honeypot niskointeraktywny skutecznie udaje przeglądarkę i umie wykryć zagrożenie, a strona działa poprawnie i rzeczywiście jest złośliwa. Autorzy malware'u stosują różne rozwiązania komplikujące analizę. Strona może np. serwować złośliwy kod jedynie ofiarom z wybranego kraju, czy też tylko co piątemu odwiedzającemu, mogą też tworzyć listy adresów IP, którym złośliwego kodu się nie serwuje – wykorzystywane ich zdaniem przez klienckie honeypoty. Strony te mogą być identyfikowane statystycznie w konsekwencji zbyt częstych odwiedzin złośliwych stron, albo poprzez wpadnięcie w pułapkę – odwiedzenie strony, o której informacja pojawiła się tylko w jednym pseudo-spamie, wysłanym świadomie na ustalony w dowolny sposób adres spamboxa wykorzystywanego do zbierania danych. Nigdy zatem nie można mieć całkowitej gwarancji poprawności wyników.

3 TECHNIKI WYKRYWANIA INFEKCJI

Podstawowym problemem w konstrukcji wysokointeraktywnego klienckiego honeypota jest konieczność automatycznego stwierdzenia, czy po odwiedzeniu danej strony w systemie doszło do nieuprawnionych zmian, a w efekcie podjęcie decyzji klasyfikującej adres jako złośliwy lub niegroźny. To podstawowa różnica pomiędzy systemami tej klasy, a sandboxowymi narzędziami do analizy zagrożeń. Technicznie oba typy narzędzi są dość podobne – w obu przypadkach działanie polega na wystawieniu działającego systemu na działanie potencjalnego zagrożenia i obserwacji skutków tej operacji w systemie. Cel działania jest jednak w obu przypadkach całkowicie różny, co wpływa na różnice w funkcjonalności narzędzi [1].

Rolą sandboxa jest wspomaganie ręcznej analizy przypadku, który jest już uznany za co najmniej podejrzany. Ostateczne wnioski z wyników badań wyciągnie człowiek. Narzędzie ma za zadanie przedstawić mu możliwie najbogatsze dane na temat zachowania badanej próbki.

Wynikiem pracy klienckiego honeypota musi natomiast być klasyfikacja – decyzja o tym, czy mamy do czynienia ze złośliwym oprogramowaniem. Dane udostępniane użytkownikom nie muszą zatem być równie bogate, kluczowe jest natomiast opracowanie dobrze działającego algorytmu klasyfikacji i skuteczne zebranie używanych przez niego informacji. Honeypot musi wykryć wystąpienie infekcji i odróżnić je od normalnego zachowania systemu. Niestety, wbrew pozorom jest to zadanie bardzo trudne.

Techniki stosowane do wykrywania infekcji w wysokointeraktywnych klienckich honeypotach można podzielić na trzy główne grupy: pasywną detekcję porównawczą, aktywne monitorowanie zdarzeń, oraz pełne monitorowanie aktywne.

3.1 Pasywna analiza porównawcza

Najprostszym, jak się wydaje, podejściem jest porównanie stanu systemu przed i po podejrzewanej infekcji. Zakłada się tutaj, że ewentualna infekcja będzie permanentna, tzn. oprogramowanie dokona pewnych zmian tak, aby uodpornić się na rutynowy reset systemu. Zmiany takie muszą oczywiście zostać wprowadzone w systemie plików – w szczególności gdzieś musi zostać zapisany plik wykonywalny, którego uruchomienie będzie wznawiało działanie infekcji.

Zaletą honeypotów używających tej metody jest ich prostota – w zasadzie możliwe jest wykorzystanie systemu bez żadnych modyfikacji, jako że podczas odwiedzania podejrzanego adresu nie wykonuje się żadnych czynności związanych z wykrywaniem infekcji. Dopiero po zakończeniu odwiedzin system jest wyłączany, a zawartość jego dysku porównywana jest z wcześniej przygotowaną kopią. Oczywiście różnice są znajdowane niemal przy każdej próbie – niektóre zmiany, np. w katalogach z ciasteczkami przeglądarek, czy ich plikami tymczasowymi, są jak najbardziej legalne, jednak stosunkowo proste jest opracowanie listy katalogów, które powinny być ignorowane, jak i listy katalogów podwyższonego ryzyka.

Inna zaleta tego podejścia to fakt, że analiza odbywa się całkowicie poza systemem wykorzystywanym jako honeypot, czyli nie ma możliwości świadomego czy przypadkowego przeszkadzania przez malware w detekcji. Malware nie ma też w zasadzie możliwości stwierdzenia, że działa w środowisku honeypota – może co najwyżej rozpoznać, że system, na którym się uruchomił, jest wirtualizowany, jednak i to ograniczenie można pokonać, jako że to podejście jest bardzo łatwe do zastosowania z fizycznymi maszynami

honeypotów. Jedynym elementem, który może zostać wykryty „od wewnątrz" jest mechanizm automatycznego otwierania badanej strony. Istnieje jednak wiele możliwych sposobów realizacji tego zadania, a żaden z nich nie wymaga ingerencji w sam system.

Mimo niezaprzeczalnych zalet, podejście pasywne nie jest zbyt często wykorzystywane – w praktyce znacznie istotniejsze okazały się jego poważne wady, z których największą jest koszmarna wręcz wydajność. Współczesne systemy wymagają dość dużych systemów plików. Konieczność zatrzymania po każdym teście maszyny wirtualnej i jej ponownego odpalenia, a w międzyczasie albo porównania kopii, albo utworzenia kopii do porównania równolegle z następnym skanem, ostatecznie w końcu przywrócenia stanu początkowego bardzo ogranicza osiągi – pojedyncza maszyna (czy to wirtualna, czy fizyczna) skanuje w tempie kilku minut na jedną odwiedzaną stronę.

Jednocześnie informacje dostarczane przez honeypoty tego typu są stosunkowo ubogie. Brak w szczególności jakichkolwiek danych na temat uruchamianych w systemie procesów. Powoduje to trudności w analizie zgłaszanych infekcji – często trudno stwierdzić, czy mamy do czynienia z fałszywym alarmem.

Obecnie trudno wskazać jakiekolwiek powszechnie znane i nadal rozwijane rozwiązanie stosujące tę technikę. Kilka lat temu była ona podstawą honeypota MITRE HoneyClient.

3.2 Aktywne monitorowanie zdarzeń

Alternatywą dla porównywania całych obrazów dysków może być rejestracja zmian w chwili, kiedy te następują. Podejście takie jest nadal stosunkowo proste w realizacji, jednak wymaga modyfikacji systemu używanego w roli honeypota. Konieczne modyfikacje, realizowane przez instalację własnych bibliotek/sterowników oraz własnych aplikacji pomocniczych, muszą zapewnić co najmniej przechwycenie wywołań interesujących funkcji systemowych (w szczególności chodzi o tworzenie i zapisywanie plików, modyfikacje rejestru i uruchamianie/kończenie procesów, ale listę można rozbudować), oraz przekazanie ich poza maszynę honeypota do dalszej analizy. W praktyce realizacja tego wymagania jest niemożliwa – zdarzeń wymienionych rodzajów może w systemie zachodzić zbyt wiele, by można je było dostatecznie szybko przekazywać na zewnątrz. Tymczasem interesująca jest jedynie niewielka ich część. Konieczne staje się filtrowanie zbieranych informacji już po stronie maszyny honeypota.

Zaletą podejścia aktywnego jest dobra wydajność. Po pierwsze, nie ma już potrzeby przeprowadzania porównania całych systemów plików – lista zarejestrowanych zdarzeń pozwala zidentyfikować dokładne położenie plików, w których zaszły zmiany. Po drugie, w przypadku, gdy system nie stwierdził infekcji, możliwe jest ponowne wykorzys-

tanie działającej już maszyny do analizy kolejnego przypadku, bez tracenia czasu na jej regenerację, a nawet restart. Oczywiście jest to obarczone pewnym ryzykiem – przegapienie infekcji w czasie pierwszego skanowania może doprowadzić do wykrycia jej w trakcie jednego z następnych, mimo że w rzeczywistości badany przypadek nie będzie złośliwy. Najczęściej zakłada się, że bez regeneracji maszyny można sprawdzić najwyżej kilka kolejnych przypadków, po przekroczeniu pewnego progu regeneracja maszyny jest wymuszana, nawet jeśli nie stwierdzono żadnej infekcji. Tak czy inaczej, osiągana w tym przypadku przeciętna wydajność jest o rząd wielkości lepsza, niż przy podejściu pasywnym – na skanowanie bezpiecznego przypadku wystarcza kilkanaście – kilkadziesiąt sekund, przypadek złośliwy również sprawdzany jest szybciej, niż w podejściu pasywnym, typowo poniżej dwóch minut.

Podejście aktywne ma oczywiście również wady. Działanie honeypota jest łatwiejsze do wykrycia. Wprawdzie i tu można stosować fizyczne maszyny, broniąc się przed wykrywaniem wirtualizacji, jednak ślad jest bardzo wyraźny – podmienione zostały przecież podstawowe systemowe biblioteki. Co gorsza, pewna kluczowa część procesu wykrywania zagrożenia dzieje się po stronie zarażonego systemu – tam odbywa się co najmniej część filtrowania informacji i przekazywanie jej na zewnątrz. Działania te można zatem zakłócić. W rozwiązaniach zamkniętych, lub zwyczajnie mało popularnych, zagrożenie świadomym zakłócaniem pracy jest niewielkie, jednak zawsze należy liczyć się z tym, że zainstalowany malware może spowodować niestabilność systemu, problemy z niektórymi funkcjami (np. przekazywaniem informacji na zewnątrz), zawieszenie lub zabicie procesu kluczowego dla analizy, a nawet załamanie systemu. W takim przypadku mamy do czynienia z trudną sytuacją decyzyjną – czy zarejestrowana awaria jest skutkiem faktycznej awarii, przypadkowych problemów w systemie, czy działania malware'u? Awaria może zresztą uniemożliwić przekazanie ostatnich, prowadzących do niej zdarzeń.

Nawet przy bezawaryjnym działaniu, nie można być całkowicie pewnym, że każde istotne zdarzenie zostanie zarejestrowane. Może zostać błędnie odfiltrowane, zgubione np. z przyczyn wydajnościowych, gdy zabraknie czasu na jego przekazanie, a w końcu może nigdy nie zostać zarejestrowane. Trudno przecież w rozbudowanym systemie operacyjnym osiągnąć absolutną pewność, że wszystkie bez wyjątku sposoby realizacji pewnego zadania (np. zapisanie pliku na dysku) zostały przechwycone.

Szczególnie dokuczliwym problemem jest niski, wywołaniowy poziom zbieranych zdarzeń. Nie są one powiązane przyczynowo, ich ilość jest duża, a większość z nich jest całkowicie normalna. Zważywszy, że ta sama akcja może być elementem infekcji, jak i na przykład automatycznej aktualizacji oprogramowania, trudno uniknąć fałszywych

alarmów, nie ograniczając wykrywalności infekcji niemal do zera. Do tego tematu jeszcze wrócimy, jako że stanowi on główny wątek artykułu.

3.3 Pełne monitorowanie aktywne

Najpoważniejsze problemy aktywnego monitorowania zdarzeń sprowadzają się do niekompletności zbieranych informacji oraz trudności z ustaleniem, które z rejestrowanych zdarzeń są objawami zachowań niepożądanych. Wad tych można uniknąć odchodząc od monitorowania pojedynczych, oderwanych od kontekstu zdarzeń w stronę pełnej analizy działania systemu. Podejście to łączy się z wykorzystaniem zaawansowanych technik analizy działania systemu, między innymi *taint tracking*, czyli śledzenia wykorzystania pamięci przez różne procesy. Monitorując całość działań systemu można zauważyć zjawiska bezpośrednio i jednoznacznie świadczące o infekcji – nadpisywanie przez program pamięci, która następnie jest traktowana jako kod do wykonania, czy też przepływ danych między programami, które w normalnych warunkach nie komunikują się.

Monitorowanie w tym trybie może odbywać się w pełni „od zewnątrz" systemu, nie da się jednak połączyć go z wykorzystaniem fizycznych maszyn. Możliwa jest zatem ochrona przed detekcją przez wykrywanie działania w warunkach debugowania/wirtualizacji.

Pełne monitorowanie aktywne nie jest aż tak kosztowne czasowo, jak mogłoby się wydawać. Oczywiście jest to wariant wolniejszy od prostego monitorowania zdarzeń, jednak bynajmniej nie wolniejszy od monitorowania pasywnego. Tak duży poziom kontroli nad działaniem systemu pozwala też na cofanie niektórych zmian, co ogranicza potrzebę restartu monitorowanego systemu. Praktycznie nie ma też w tym przypadku problemu fałszywych alarmów – alarm jest zawsze dobrze uzasadniony i świadczy o wystąpieniu nieprawidłowych zachowań, co najwyżej w rzadkich wypadkach zachowania te mogą być celowe i pozbawione negatywnych skutków.

Najpoważniejszą wadą pełnego monitorowania jest trudność implementacji. O ile honeypoty wykorzystujące pozostałe techniki są rozwiązaniami stosunkowo prostymi, sprowadzającymi się do automatyzacji pracy systemu i dodania niezbyt skomplikowanych mechanizmów detekcji (pomińmy na chwilę trudny temat filtrowania uzyskiwanych informacji, czyli podejmowania decyzji o alarmie), o tyle honeypot z pełnym monitorowaniem jest kompletnym rozwiązaniem sandboxowym wykorzystującym bardzo zaawansowane techniki.

Pierwszym klienckim honeypotem wykorzystującym tę technikę był opracowany na Vrije Universiteit Amsterdam system Shelia[2], opisany w pracy [2]. Obecnie jego rozwój został zarzucony, a autorzy skupili się na dodaniu funkcji klienckiego honeypota do systemu sandboxowego Argos[3], aby nie utrzymywać równolegle dwóch bardzo w gruncie rzeczy podobnych rozwiązań. Funkcjonalność Argosa [3] może znacznie wzbogacić uzyskiwane dane w porównaniu do Shelii, choć można się zastanawiać, czy jest to istotne w zastosowaniu honeypotowym, gdzie istotna jest raczej sama decyzja o alarmie – dane można zebrać później.

3.4 Podsumowanie

Powyższy opis mógłby sugerować, że techniki monitorowania tworzą prostą hierarchię i wybór jest trywialny – jeśli tylko jest to technicznie możliwe, należy stosować pełne monitorowanie. W rzeczywistości decyzja nie jest tak prosta. Prostota i wydajność aktywnego monitorowania zdarzeń ma swoje zalety. Oprogramowanie stosujące to podejście można łatwo modyfikować i dostosować do współpracy z dowolnym innym wykorzystywanym oprogramowaniem. Tymczasem żadne z proponowanych podejść nie jest w stanie rozwiązać podstawowego problemu wysokointeraktywnych klienckich honeypotów, jakim są pominięte alarmy. Honeypot może wykryć jedynie zagrożenie, na które jest podatny, a to oznacza, że niezależnie od techniki monitorowania kluczowy jest wybór wersji oprogramowania instalowanych w monitorowanym systemie, a także możliwość jednoczesnego wykorzystywania więcej niż jednej konfiguracji. Nic zatem dziwnego, że o ile wykrywanie pasywne zostało w zasadzie zarzucone, obie pozostałe metody są wykorzystywane w praktyce.

Dalsza część artykułu skupia się na problemie fałszywych alarmów występujących w systemach opartych na aktywnym monitorowaniu zdarzeń.

4 CAPTURE-HPC

Capture-HPC[4] jest wysokointeraktywnym klienckim honeypotem opartym na aktywnym monitorowaniu zmian w systemie. Jest to jeden z najstarszych projektów tego typu o otwartym kodzie źródłowym. System stworzony przez Christiana Seiferta w ramach pracy doktorskiej wzbudził zainteresowanie członków Honeynet Project i jest obecnie przez nich wspierany. Jednocześnie – i niezależnie – został on wybrany do wykorzysta-

[2] http://www.cs.vu.nl/~herbertb/misc/shelia/
[3] http://www.few.vu.nl/argos/
[4] https://projects.honeynet.org/capture-hpc

nia w roli modułu wysokointeraktywnego w projekcie HoneySpider Network[5], prowadzonym przez NASK, GOVCERT.NL i SurfNet [4]. Projekt ten miał na celu stworzenie systemu klienckich honeypotów przetwarzającego w sposób zautomatyzowany bardzo duże ilości adresów URL pochodzących z różnych źródeł, wykorzystującego analizę zarówno nisko- jak i wysokointeraktywną. Stworzony system jest obecnie wykorzystywany operacyjnie w należącym do NASK zespole CERT Polska, a także w innych miejscach na świecie, trwają prace nad drugą wersją systemu. Dalszy tekst opiera się głównie na doświadczeniach autora z wersją Capture-HPC zmodyfikowaną w ramach tego właśnie projektu, jednak obie wersje są bardzo zbliżone.

4.1 Ogólna struktura systemu

System Capture-HPC składa się z dwóch modułów, z których jeden pracuje wewnątrz maszyny wirtualnej (tzw. klient), a drugi poza nią (tzw. serwer). System wykorzystuje istniejące technologie wirtualizacji – w wersji oryginalnej VMWare, a w wersji działającej w projekcie HoneySpider Network – VirtualBox. Komunikacja między serwerem a klientami odbywa się poprzez sieć.

Serwer jest napisany w Javie i odpowiada za zarządzanie przetwarzaniem URLi. Jeden serwer może zarządzać kilkoma maszynami wirtualnymi, komunikując się z zainstalowanymi na nich klientami, przydzielając im adresy do zbadania i decydując o restarcie i regeneracji maszyn.

Klient jest modułem odpowiedzialnym za właściwą detekcję infekcji. Składa się z zestawu sterowników, które podmieniają odpowiednie funkcje systemowe, oraz aplikacji, która komunikuje się z serwerem, uruchamia przeglądarkę z odpowiednim adresem, zbiera zdarzenia zarejestrowane przez sterowniki, filtruje je, aby pozbyć się fałszywych alarmów, zbiera inne dane, a w końcu przekazuje całość informacji do serwera.

Dane zbierane przez klienta to:

- zdarzenia systemowe (podlegają filtrowaniu):
  - zmiany w rejestrze,
  - zmiany w systemie plików,
  - uruchomienia i zatrzymania procesów,
- modyfikowane, tworzone i kasowane pliki,
- całość ruchu sieciowego (z wyjątkiem ruchu między klientem a serwerem).

Ponieważ zdarzenia normalne w zdrowym systemie są odfiltrowywane przez klienta, klasyfikacja adresu przez serwer jest bardzo prosta – jeśli klient przekazał jakiekolwiek

[5] http://www.honeyspider.net/

zarejestrowane zdarzenia, to adres jest złośliwy. Oznacza to, że całkowita odpowiedzialność za jakość klasyfikacji spada na część kliencką.

4.2 Listy wykluczeń

Filtrowanie zdarzeń w kliencie odbywa się poprzez porównywanie odebranych zgłoszeń z przygotowanymi listami wykluczeń. Są to listy wyrażeń regularnych, specyfikujących wzorce użyteczne w klasyfikacji. Większość pozycji na listach to – zgodnie z nazwą – wykluczenia, tzn. dowolne zdarzenie pasujące do wzorca jest traktowane jako poprawne i ignorowane. Istnieje jednak również możliwość definiowania wzorców pozytywnych, tzn. dowolne zdarzenie pasujące do takiego wzorca będzie uznane za złośliwe, niezależnie od tego, czy pasuje też do jakiegoś wzorca wykluczającego. Pozwala to definiować wyjątki, co znacznie upraszcza tworzenie czytelnych i efektywnych wyrażeń regularnych. Niestety nie ma możliwości tworzenia bardziej skomplikowanych, wielopoziomowych struktur wyjątków (wyjątek od wyjątku, itp.).

Definicja list wykluczeń jest bardzo ważnym, trudnym i pracochłonnym zadaniem w ramach implementacji i utrzymania honeypota stosującego technikę aktywnego monitorowania zdarzeń. Już pojedyncze pominięte legalne zdarzenie spowoduje fałszywy alarm, tymczasem stworzenie wzorca zbyt ogólnego odfiltruje też zdarzenia nielegalne, obniżając skuteczność detekcji. Listy wykluczeń dołączone do oryginalnego Capture-HPC są w polskich warunkach praktycznie bezużyteczne – zbyt wiele ścieżek różni się w wyniku różnicy językowej. Wprowadzenie odpowiednich zmian nie wystarcza – listy są stosunkowo ubogie. Co więcej, zestaw spotykanych w poprawnie działającym systemie zdarzeń może się zmieniać przy każdej aktualizacji systemu lub zainstalowanych w nim aplikacji.

Specyfikacja listy wykluczeń jest zatem pracochłonnym, ręcznym zadaniem, polegającym na obserwacji działania systemu w normalnych warunkach, rejestrowaniu zachodzących zdarzeń i dodawaniu wykluczających wzorców aż do osiągnięcia regularnych, „cichych” przebiegów, w których wszystkie zdarzenia zachodzące przy odwiedzaniu zwykłych stron są odfiltrowywane. Dodając kolejne linijki szczególną uwagę należy poświęcić formułowaniu wyrażeń regularnych – powinny pasować do wszystkich zdarzeń podobnych do złapanego, ale nie więcej. Wyzwanie polega na takim formułowaniu wyrażeń, aby zadanie wykonać możliwie niewielką ich liczbą. W przeciwnym razie pojawia się zagrożenie przeciążeniem maszyny – klient będzie tak wiele czasu poświęcał na porównywanie zdarzeń z wzorcami, że nie będzie w stanie odbierać nadchodzących zgłoszeń. Niestety, jest to poszukiwanie złotego środka – nie da się osiągnąć jednocześnie pewności odfiltrowania wszystkich legalnych zdarzeń, pewności zgłoszenia wszyst-

kich zdarzeń nielegalnych i odpowiednich rozmiarów list wykluczeń. W praktyce nawet dla bardzo dobrze zestrojonych list wykluczeń często zdarzają się fałszywe alarmy, a niektóre zdarzenia są gubione.

5 UNIKANIE FAŁSZYWYCH ALARMÓW

Problem fałszywych alarmów w systemie Capture-HPC wynika z niedoskonałości mechanizmu list wykluczeń. Fałszywy alarm jest bowiem zawsze skutkiem przekazania serwerowi przez moduł klienta zdarzeń, które nie są w istocie złośliwe. Niestety nie jest to jedynie skutek trudności jednoznacznego sklasyfikowania pojedynczego zdarzenia. Filtrowanie po stronie klienta stosowane jest do każdego zdarzenia osobno, tymczasem złośliwość niektórych zdarzeń może być uzależniona od całej sekwencji zdarzeń wcześniejszych i późniejszych. Nawet zatem gdyby udało się opracować listy doskonałe (co jest raczej niemożliwe), nie mogłyby one być w pełni skuteczne.

Tymczasem fałszywy alarm jest w wielu zastosowaniach nawet bardziej szkodliwy, niż błędne sklasyfikowanie strony jako niegroźnej. Dla przykładu, zespół typu CSIRT[6] obsługujący duży segment sieci, taki jak CERT Polska, w oczywisty sposób nie jest w stanie zidentyfikować i obsłużyć wszystkich występujących w tej sieci zagrożeń – wymagałoby to gigantycznego budżetu i setek ludzi. Tymczasem ręczne sprawdzenie każdego fałszywego alarmu zajmuje czas – nawet przy olbrzymim doświadczeniu i oczywistym przekła-maniu jest to co najmniej kilka, a raczej kilkanaście sekund. Już pojedyncza instancja Capture-HPC, wykorzystująca np. 6 maszyn wirtualnych, jest w stanie przetwarzać wiele tysięcy adresów dziennie. Nawet umiarkowana stopa błędów może więc skutkować dziesiątkami, a nawet setkami fałszywych alarmów, czyli całymi godzinami zmarnowanego czasu jednej osoby.

Rozwiązanie zaproponowane jako rozszerzenie modułu Capture-HPC w systemie HoneySpider Network, opracowane w ramach projektu WOMBAT[7], opiera się na prostym pomyśle post-procesora [5]. W zwykłym honeypocie Capture-HPC fakt istnienia niepustego logu ze zdarzeniami jest wystarczającą podstawą uznania strony za złośliwą. Aby ograniczyć liczbę fałszywych alarmów, wprowadzono dodatkowy moduł eliminatora fałszywych alarmów, który przejmuje rolę klasyfikatora. Brak zdarzeń w logu dla danej strony w dalszym ciągu interpretowany jest jednoznacznie jako dowód na niezłośliwość strony, ale istnienie logu nie jest już jednoznaczne – mogą istnieć logi niezłośliwe.

[6] Computer Security Incident Response Team

[7] *Worldwide Observatory of Malicious Behaviors and Attack Threats*, projekt 7-go programu ramowego UE nr FP7-ICT-216026, http://www.wombat-project.eu.

Takie podejście wydaje się dość intuicyjne, jednak pozostaje pytanie, jak realizować klasyfikację logów. Można oczywiście oprzeć się na regułach, bazujących na wyrażeniach regularnych. Jest to niewielkie rozszerzenie funkcjonalności w porównaniu z filtrowaniem po stronie klienta – zaletą jest umieszczenie filtrowania poza maszyną wirtualną i jej ograniczeniem czasowym na przekazanie wyniku, ponadto możliwe jest uwzględnienie złośliwości lub niezłośliwości pewnych większych sekwencji zdarzeń. Prace konfiguracyjne stają się jednak przez to jeszcze bardziej złożone i trudne – trzeba opracowywać dwie różne listy wykluczeń. Można też mieć wątpliwości, czy w ten sposób da się skutecznie ograniczyć liczbę błędów.

Wybór padł na rozwiązanie mniej przewidywalne, ale rokujące większe nadzieje – uczenie maszynowe, czyli techniki sztucznej inteligencji. Klasyfikacja następuje na podstawie kombinacji cech liczonych dla każdego logu, ale zasada podejmowania decyzji nie jest dana wprost, lecz wyuczona na podstawie zbioru uczącego. Zbiór ten składał się z ponad 80 logów, sklasyfikowanych ręcznie jako złośliwe przez specjalistę z działu CERT Polska, oraz kilkaset logów zdaniem specjalisty niezłośliwych. Dopiero tak nauczony system gotowy jest do klasyfikowana nowych przykładów.

5.1 Przygotowanie mechanizmu klasyfikacji

Wbrew potocznej opinii, algorytmy uczenia maszynowego nie działają na zasadzie czarnej skrzynki, czyli całkowicie automatycznej klasyfikacji obiektów (porcji danych) według podanych przykładów. Klasyfikacja odbywa się na podstawie cech obiektu, czyli pewnych jego istotnych właściwości. Cechy te musi wskazać twórca systemu uczącego się. Uzasadnienie tego faktu jest bardzo intuicyjne: przyjmijmy, że system ma klasyfikować pliki binarne. Automat nie jest w stanie sam stwierdzić, co jest istotne. Wartość osiemnastego bajtu w pliku? Wartość bitu, którego położenie w pliku wskazują pierwsze bajty? A może cały plik jest zawsze obrazkiem i istotny jest jego format i rozmiar? A może to jest obrazek, ale format może być różny i jest bez znaczenia, a decydują proporcje między barwą czerwoną a niebieską w kwadracie 5 na 5 pikseli w prawym górnym rogu obrazka? Możliwości jest nieskończenie wiele. Ostatecznie zatem to właśnie wybór cech decyduje o skuteczności uczenia maszynowego. Algorytm sam ustali, które z cech są ważniejsze, jakie ich kombinacje wskazują na przynależność obiektu do określonej kategorii, ale sam zbiór cech musi być podany z góry. Pominięcie w nim cech znaczących, mających duży wpływ na klasyfikację, może uniemożliwić programowi skuteczną klasyfikację. Przesada w drugą stronę, czyli definiowanie wszelkich cech, jakie tylko przyjdą do głowy projektantowi, jest równie niekorzystne – po pierwsze, na tak dużym zbiorze cech nie da się wydajnie operować, zarówno uczenie, jak i sama klasy-

fikacja, będą bardzo powolne; po drugie istnieje ryzyko „przeuczenia" algorytmu – można przypadkowo wskazać cechy, które akurat na zbiorze uczącym bardzo dobrze sprawdzają się w klasyfikacji, natomiast całkowicie zawodzą w ogólniejszym przypadku.

Można tu przywołać znaną anegdotę o przeuczeniu systemu do rozpoznawania zamaskowanych czołgów na obrazkach. System wykazywał doskonałą jakość klasyfikacji na zbiorze testowym, jednak w praktyce zawodził. Po analizie wyników okazało się, że zarówno w zbiorze uczącym, jak i testowym, wszystkie zdjęcia faktycznie przedstawiające czołgi były wykonane w lecie na poligonie, natomiast pozostałe pochodziły z różnych miejsc i pór roku. Algorytm znalazł z łatwością rzeczywiście najskuteczniejszy dla tych zbiorów wariant – wybierał zdjęcia najbardziej zielone… Opracowano skuteczny system wykrywania trawy.

Wybór cech istotnych w zastosowaniu do klasyfikacji logów zwracanych przez Capture-HPC był prawdopodobnie najważniejszą częścią przedsięwzięcia. Wybrane należały do trzech głównych grup:

- Cechy wierszy – określają cechy pojedynczych wierszy w logu i mogą być stosowane częściowo niezależnie od pozostałych. Jeśli program przy użyciu tych cech jednoznacznie określi wszystkie wiersze jako legalne, to log można uznać za niezłośliwy, bez potrzeby dalszego rozpatrywania. Także odwrotnie, wystąpienie nawet jednej jednoznacznie złośliwej linijki wystarcza, aby uznać log za złośliwy. Ta grupa cech jest wykorzystywana w trybie najbardziej przypominającym działanie list wykluczeń i w pewnym sensie uzupełnia niedostatki tych list.
- Cechy zbiorowe logów – określają cechy całego logu, traktowanego jako nieuporządkowany zbiór wierszy. Na przykład, cechą logu może być to, ile razy wystąpiła w nim operacja uruchomienia procesu.
- Cechy sekwencyjne logów – określają cechy całego logu, traktowanego jako uporządkowany zbiór wierszy. Na przykład, cechą logu może być to, czy występuje w nim operacja uruchomienia procesu z pliku, do którego wcześniej w tym samym logu zarejestrowano operację zapisu.

Obszerne testy z wieloma cechami ze wszystkich trzech grup doprowadziły do stworzenia niewielkiego zestawu cech, które dobrze sprawdzają się w klasyfikacji logów. Szczegóły można znaleźć w opracowaniu [5].

Przeprowadzono dużą liczbę testów z wykorzystaniem dostępnych, ręcznie sklasyfikowanych logów, jak i dużych pul niesklasyfikowanych wcześniej adresów. Na tej podstawie dokonano wyboru algorytmu klasyfikacji.

5.2 Skuteczność rozwiązania

Przeprowadzone testy zaimplementowanego rozwiązania pokazały, że klasyfikację można prowadzić praktycznie w czasie rzeczywistym – wydajność rzędu 1200 logów na minutę jest o całe rzędy wielkości wyższa, niż wydajność wysokointeraktywnych klienckich honeypotów, pojedynczy analizator logów jest w stanie obsłużyć jednocześnie wiele instancji Capture-HPC.

Ważniejszym jednak zadaniem jest ocena skuteczności rozwiązania. Tu warto zwrócić uwagę na jeden z przeprowadzonych testów, w którym badano adresy z dużej puli, zawierającej niewielki odsetek rzeczywiście złośliwych. Testy przeprowadzono z wykorzystaniem wyników dwóch różnych instalacji Capture-HPC. Jedna z nich używała eksperckich list wykluczeń, przygotowanych na użytek HoneySpider Network, druga natomiast – domyślnych list wykluczeń pakietu Capture-HPC, zmodyfikowanych jedynie przez tłumaczenie nazw katalogów dla zgodności z polską wersją Windows XP. Ten drugi wariant miał umożliwić ocenę, czy stworzony klasyfikator mógłby wyeliminować wysiłek potrzebny na tworzenie dobrych list wykluczeń, ograniczając je do zadania odfiltrowania najczęstszych, w najbardziej oczywisty sposób legalnych zdarzeń w celu zmniejszenia rozmiarów logów, a więc i przyspieszenia ich przesyłu z klienta do serwera. W Tabeli 1 przedstawiono wyniki dla trzech różnych grup adresów.

Tabela 1. Wyniki testów automatycznej klasyfikacji

Rodzaj wyniku	Zbiór 1		Zbiór 2		Zbiór 3	
	Listy eksperckie	**Listy domyślne**	**Listy eksperckie**	**Listy domyślne**	**Listy eksperckie**	**Listy domyślne**
Liczba logów Capture-HPC	164	15152	172	12824	183	12862
w tym ręcznie potwierdzonych	0	b.d.	4	b.d.	4	b.d.
Liczba logów uznanych za złośliwe	0	277	4	13	3	9
w tym ręcznie potwierdzonych	0	31	1	0	3	1

Jak widać w tabeli, skuteczność zaproponowanego rozwiązania jest bardzo wysoka – redukuje ono liczbę fałszywych alarmów wielokrotnie, nie tracąc przy tym zbyt wielu przypadków potwierdzonych. Dla list eksperckich algorytm sprawdził się doskonale na pierwszym zbiorze testowym, na trzecim odrzucił omyłkowo jeden log, ale nie wygenerował ani jednego fałszywego alarmu, a na drugim jego wynik, choć najsłabszy, jest

nadal dobry dla przewidywanego zastosowania – wygenerowano 3 fałszywe alarmy i jeden prawdziwy, choć powinno być 4. Równie zadowalający jest wynik dla domyślnych list wykluczeń – dla zbioru pierwszego osiągnięto liczbę alarmów tego samego rzędu, co przy listach eksperckich, a dla pozostałych zbiorów znacząco mniejszą. Uzyskano też prawdziwe alarmy. Można zatem liczyć na to, że uczeniem maszynowym można w pewnej mierze zastąpić dokładne strojenie list, choć osiągnięcie jakości nieco lepszej, niż dla list domyślnych, byłoby warte zachodu.

Przy interpretacji tabeli należy pamiętać, że rzeczywista liczba stron złośliwych w obu wariantach dla tego samego zbioru nie musi być równa – są to wyniki dwóch różnych skanowań, różna mogła być dostępność poszczególnych adresów, na niektórych stronach malware mógł być serwowany niedeterministycznie (np. co drugiemu odwiedzającemu), ponadto maszyny wirtualne w obu wariantach nie były identycznymi kopiami – wersje niektórych programów, a więc i wrażliwość na pewne zagrożenia, mogły się różnić.

Pojedyncza instalacja HoneySpider Network, wyposażona tylko w jeden moduł wysokointeraktywny, pracujący na kilku maszynach wirtualnych, skanująca non-stop adresy z typowych źródeł i wykorzystująca listy eksperckie wytwarza logi w tempie rzędu 4000 dziennie. Nawet 10 sekund analizy każdego przypadku (przy założeniu, że większość alarmów jest fałszywa) wymagałoby 11 godzin pracy specjalisty dziennie – półtora etatu. Tymczasem zastosowanie automatycznej klasyfikacji zmniejsza przeciętną liczbę alarmów do kilkudziesięciu, może stu kilkudziesięciu, co wymaga już tylko kilkunastu minut. Można zatem badać każdy przypadek dokładniej, oddelegować specjalistów do bardziej wymagających zadań, lub zbudować znacznie większą instalację do detekcji.

6 PODSUMOWANIE

Niniejszy artykuł miał za zadanie pokazać czytelnikowi, na czym polega działanie wysokointeraktywnych klienckich honeypotów. Wyjaśniono, czym różnią się od honeypotów serwerowych i niskointeraktywnych. Zaprezentowano różne techniki wykrywania infekcji honeypota. Przedstawiona została też koncepcja działania honeypota Capture-HPC. Wymienione wyjaśnienia powinny dostatecznie tłumaczyć fakt powstawania wielu fałszywych alarmów. Wyjaśniono, dlaczego – o ile honeypot pracuje w zespole reagowania jako źródło informacji – są one nawet bardziej dotkliwe od fałszywych pominięć alarmów. Ostatecznie pokazano jeden ze sposobów radzenia sobie z tym problemem, wykorzystywany w module zaprojektowanym w NASK. Celem artykułu jest uświadomienie czytelnikom poziomu trudności zadania zbierania danych do działalności operacyjnej dużego zespołu reagowania na incydenty.

6.1 Uwagi końcowe

Choć w artykule opisano kilka rozwiązań opracowanych w NASK, w których tworzeniu autor brał udział, nie należy zakładać, że są to wyłączne dzieła autora. W szczególności projekt HoneySpider Network rozwijany był przez liczny zespół, kierowany przez Piotra Kijewskiego z CERT Polska. Badania, projekt i implementacja opisywanego w ostatnim rozdziale modułu eliminacji fałszywych alarmów są natomiast w największej mierze dziełem Cezarego Żukowskiego z Zespołu Metod Bezpieczeństwa Sieci i Informacji.

Literatura

1. Kozakiewicz, A.(red. nauk.): *WOMBAT project public deliverable D23/D5.3 Early Warning System*: Experimental report, http://wombat-project.eu/-WP5/FP7-ICT-216026-Wombat_WP5-_D23_V01_Early-warning-system-experimental-report.pdf.
2. Rocaspana, J. R.: *Shelia: A Client Honeypot for Client-Side Attack Detection*, dokumentacja projektu, http://www.cs.vu.nl/~herbertb/-misc/shelia/shelia-07.pdf.
3. Portokalidis, G., Słowińska, A., Bos, H.: *Argos: an Emulator for Fingerprinting Zero-Day Attacks*, Proc. ACM SIGOPS EUROSYS'2006, Leuven, Belgia, 2006.
4. Kijewski, P., Overes, C., Spoor, R.: *The HoneySpider Network – fighting client side threats*, 20th Annual FIRST Conference on Computer Security Incident Handling, Vancouver, Kanada, 2008.
5. Comparetti, P. M. (red. nauk.): *WOMBAT project public deliverable D21/D4.7* Consolidated report with evaluation results, http://wombat-project.eu/WP4/FP7-ICT-216026-Wombat_WP4_D21_V01_Consolida-ted-reports-with-evaluation-results.pdf.