Charakterystyka Kippo z punktu widzenia integracji honeypot'a z innymi systemami informatycznymi

Kamil Kołtyś

Pracownia Metod Bezpieczeństwa Sieci i Informacji

Spis treści

1 Wstęp

Kippo jest honeypot'em emulującym usługę SSH. Emulacja obejmuje procedurę nawiązania sesji SSH, a także interakcję z powłoką systemową w przypadku pomyślnego uwierzytelnienia użytkownika. Honeypot może być zastosowany do rejestrowania ataków typu "brute force" mających na celu złamanie hasła danego użytkownika do usługi SSH. Ponadto pozwala on na analizę dalszych działań atakującego w przypadku, gdy atakujący poda prawidłowe hasło i uzyska dostęp do emulowanego systemu. Kippo rejestruje bowiem wszystkie polecenia powłoki wprowadzane przez użytkownika, a także zapisuje we wskazanym katalogu pliki pobierane przez niego za pomocą emulowanego polecenia *wget*.

Kippo na tyle dokładnie emuluje usługę SSH, że atakujący może mieć wrażenie, iż korzysta z rzeczywistej usługi SSH. Dzięki temu dane zbierane przez honeypot'a dostarczają wiele ciekawych informacji dotyczących celu i sposobu przeprowadzania tego typu ataków. Takie dane mogą być użyteczne dla różnych systemów informatycznych, zwłaszcza tych związanych z bezpieczeństwem teleinformatycznym. Celem niniejszego opracowania jest scharakteryzowanie Kippo z punktu widzenia możliwości jego integracji z innym systemem, który dzięki funkcjonalności honeypot'a będzie mógł lepiej realizować swoje zadania w dziedzinie bezpieczeństwa teleinformatycznego.

W kolejnych rozdziałach omówiono najpierw kwestie związane z instalacją, konfiguracją oraz uruchomieniem i ewentualną zmianą konfiguracji honeypot'a, a następnie scharakteryzowano dane zbierane przez Kippo dokładnie przedstawiając format, w jakim te dane są udostępniane na zewnątrz.

2 Instalacja

Instalacja polega skopiowaniu do wybranego katalogu, który będzie katalogiem roboczym Kippo, następujących elementów z repozytorium SVN `http://kippo.googlecode.com/svn/trunk/`:

- *kippo* – katalog z plikami źródłowymi Kippo (python);
- *kippo.tac* – plik konfiguracyjny dla *twistd* (aby uruchomić Kippo, w systemie musi być zainstalowane narzędzie *twistd*);
- *start.sh* – skrypt powłoki uruchamiający Kippo;

3 Konfiguracja

3.1 Główny plik konfiguracyjny

Zasadnicza część konfiguracji Kippo znajduje się w pliku tekstowym o formacie podobnym do INI [1], który jest obsługiwany m. in. przez moduł python *ConfigParser*. Z repozytorium Kippo można pobrać przykładowy plik konfiguracyjny `http://kippo.googlecode.com/svn/trunk/kippo.cfg.dist`.

Kippo zakłada, że ten plik ma nazwę *kippo.cfg* i znajduje się w katalogu roboczym Kippo. Nazwę i lokalizację pliku konfiguracyjnego można zmienić jedynie modyfikując kod źródłowy Kippo. W tym celu w pliku *kippo.tac* (wiersz `if not os.path.exists('kippo.cfg')`) oraz w pliku *kippo/core/config.py* (wiersz `for f in ('kippo.cfg', '/etc/kippo/kippo.cfg', '/etc/kippo.cfg'):`) zamiast `'kippo.cfg'` należy podać pełną ścieżkę do pliku konfiguracyjnego.

Plik konfiguracyjny zawiera trzy sekcje:

- [honeypot] – sekcja obowiązkowa zawierająca podstawową konfigurację honeypota;
- [database_mysql] – sekcja opcjonalna zawierająca konfigurację modułu logowania do bazy danych MySQL;
- [database_xmpp] – sekcja opcjonalna zawierająca konfigurację modułu logowania do serwera XMPP;

Poniżej omówiono wszystkie parametry poszczególnych sekcji.

Sekcja [honeypot]

W sekcji [honeypot] można ustawić następujące parametry:

- ssh_addr - adres, na którym ma nasłuchiwać honeypot (domyślnie wartość 0.0.0.0 oznaczająca każdy adres IP);
- ssh_port - port, na którym ma nasłuchiwać honeypot (domyślnie 2222);
- hostname - wyświetlana przez powłokę nazwa hosta;
- log_path - miejsce tworzenia logów (UWAGA! podany katalog musi mieć podkatalog *tty*, w którym będą zapisywane pliki z przebiegiem poszczególnych sesji SSH; ponadto miejsce tworzenia głównego logu należy odpowiednio zmodyfikować w skrypcie start.sh - patrz 3.2);

[1] Różnica polega na tym, że linię zawierającą komentarz rozpoczyna się znakiem # a nie znakiem ;.

- download_path - katalog, w którym będą zapisywane pobierane (za pomocą emulatora *wget*) pliki;
- download_limit_size - maksymalny rozmiar dla pobieranego pliku (pliki o większym rozmiarze niż podany nie będą w ogóle zapisywane; domyślnie wartość 0 oznaczająca brak limitu);
- filesystem_file – ścieżka do pliku emulującego system plików (patrz 3.3);
- contents_path – katalog z treściami plików (wyświetlanymi przez emulator *cat*) (patrz 3.4);
- txtcmds_path – katalog z komunikatami generowanymi przez pliki wykonywalne (patrz 3.5);
- data_path - katalog zawierający pliki z dodatkowymi danymi konfiguracyjnymi (patrz 3.6);
- public_key - klucz publiczny serwera SSH emulowanego przez Kippo;
- private_key - klucz prywatny serwera SSH emulowanego przez Kippo;
- out_addr - adres IP wykorzystywany przez emulatora *wget* do tworzenia wychodzących połączeń (parametr opcjonalny);
- sensor_name - nazwa identyfikująca instancję Kippo w bazie danych (parametr opcjonalny, jeżeli nie jest podany, to nazwą identyfikującą instancję Kippo w bazie danych jest adres IP hosta);
- fake_addr - sztuczny adres IP wyświetlany jako adres źródłowy połączenia SSH przez emulator polecenia *w* (parametr opcjonalny, jeżeli nie jest podany, to emulator *w* wyświetla rzeczywisty adres IP);
- banner_file – plik zawierający komunikat, który ma być wyświetlany przed pierwszym pytaniem o hasło;
- interact_enabled – flaga (true lub false) wskazująca czy interaktywny dostęp do Kippo przez telnet ma być włączony;
- interact_port – port, na którym Kippo zapewnia interaktywny dostęp przez telnet;

Sekcja [database_mysql]

W sekcji [database_mysql] można ustawić następujące parametry:

- host - nazwa serwera z bazą danych;
- database - nazwa bazy danych;
- username - nazwa użytkownika;
- password - hasło użytkownika;
- port – port, na którym należy łączyć się z bazą danych;

Sekcja [database_xmpp]

W sekcji [database_xmpp] można ustawić następujące parametry:

- server – nazwa serwera XMPP;
- user – nazwa użytkownika;
- password – hasło użytkownika;
- muc – nazwa usługi MUC;
- signal_createsession - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o utworzeniu sesji;
- signal_connectionlost - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o utracie połączenia;
- signal_loginfailed - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o próbie nieudanego logowania;
- signal_loginsucceeded - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o udanym logowaniu;
- signal_command - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o wykonaniu polecenia powłoki;
- signal_clientversion - nazwa pokoju MUC, do którego są wysyłane wiadomości informujące o wersji klienta SSH;
- debug – flaga (true lub false) wskazująca czy dodatkowe informacje dotyczące komunikacji z serwerem XMPP mają być logowane;

3.2 Parametry wywołania twistd

W skrypcie *start.sh* jako parametry wywołania *twistd* można podać:

- ścieżkę do głównego pliku z logiem Kippo (domyślnie *log/kippo.log*);
- ścieżkę do pliku z identyfikatorem PID procesu Kippo (domyślnie *kippo.pid*);

3.3 Plik emulujący system plików

Kippo wymaga wskazania pliku, który będzie wykorzystywany do emulacji systemu plików. W tym pliku znajduje się informacja o strukturze emulowanego systemu plików (bez treści samych plików).

W repozytorium Kippo jest udostępniony przykładowy plik `http://kippo.googlecode.com/svn/trunk/fs.pickle`, który emuluje system plików typowy dla systemu Debian 5.0. Niemniej za pomocą dostępnego w repozytorium Kippo narzędzia `http://kippo.googlecode.com/svn/trunk/createfs.py` można wygenerować plik, który emuluje system plików macierzystego systemu.

3.4 Treść plików

Plik emulujący system plików definiuje jedynie strukturę systemu plików i nie zawiera treści poszczególnych plików. W związku z tym emulowane przez Kippo polecenie *cat* domyślnie dla każdego pliku wypisuje komunikat No such file or directory, nawet jeżeli ten plik istnieje w strukturze emulowanego systemu plików (wyświetla go emulowane przez Kippo polecenie *ls*).

Niemniej dla danego pliku w emulowanym systemie można określić treść, która będzie wyświetlana przez polecenie *cat*. W tym celu, we wskazanym w konfiguracji katalogu (parametr contents_path) należy utworzyć plik o określonej treści (i o tej samej nazwie co dany plik) w podkatalogu odpowiadającym miejscu tego pliku w emulowanym systemie plików. Na przykład, jeżeli podanym w konfiguracji katalogiem, w którym mają być zdefiniowane zawartości plików, jest */var/kippo/honeyfs*, to emulowane przez Kippo polecenie *cat* wywołane na pliku */etc/passwd* wyświetli zawartość pliku */var/kippo/honeyfs/etc/passwd* (o ile ten plik istnieje, w przeciwnym razie zostanie zwrócony komunikat No such file or directory).

W repozytorium Kippo znajduje się katalog `http://kippo.googlecode.com/svn/trunk/honeyfs` z przykładową treścią wybranych plików systemu Debian 5.0.

3.5 Komunikaty plików wykonywalnych

W systemie emulowanym przez Kippo próba uruchomienia danego pliku wykonywanego domyślnie powoduje wyświetlenie komunikatu command not found.

Niemniej dla danego pliku wykonywalnego znajdującego się w emulowanym systemie plików można zdefiniować komunikat, który będzie wyświetlany po uruchomieniu tego pliku. W tym celu, we wskazanym w konfiguracji katalogu (parametr txtcmds_path) należy utworzyć plik o określonej treści (i o tej samej nazwie co dany plik) w podkatalogu odpowiadającym miejscu tego pliku w emulowanym systemie plików. Na przykład, jeżeli podanym w konfiguracji katalogiem, w którym mają być zdefiniowane treści plików, jest */var/kippo/txtcmds*, to wykonanie pliku */bin/mount* wyświetli komunikat zawarty w pliku */var/kippo/txtcmds/bin/mount* (o ile ten plik istnieje, w przeciwnym razie zostanie wyświetlony komunikat command not found).

W repozytorium Kippo znajduje się katalog `http://kippo.googlecode.com/svn/trunk/txtcmds` z przykładową treścią komunikatów dla wybranych plików wykonywalnych systemu Debian 5.0.

3.6 Dodatkowe dane konfiguracyjne

Na dodatkowe dane konfiguracyjne składają się dwa pliki znajdujące się w katalogu ustalonym w głównym pliku konfiguracyjnym przez parametr data_path:

- *userdb.txt* - plik tekstowy zawierający w każdym wierszu trójkę login:uid:hasło; dany login może wystąpić wiele razy w kilku różnych wierszach (trójkach) i w związku z tym może być on skojarzony z wieloma różnymi hasłami; uwierzytelnienie polega na sprawdzeniu czy dla danego loginu i hasła jest zdefiniowana w pliku *userdb.txt* trójka zawierające ten login i hasło (ewentualnie zdefiniowana treść pliku */etc/passwd* nie jest brana pod uwagę); uid skojarzony z danym loginem (w przypadku gdy login występuje w wielu trójkach, to jest mu przydzielany zawsze uid z pierwszej zdefiniowanej trójki) jest wyświetlany przez emulator polecenia *id*; zmiana hasła za pomocą emulowanego przez Kippo polecenia *passwd* powoduje dodanie na końcu pliku *userdb.txt* trójki cur_user:cur_uid:new_pass, gdzie cur_user jest nazwą zalogowanego użytkownika, cur_uid uid'em zalogowanego użytkownika, a new_pass zmienionym hasłem; przy kolejnym uwierzytelnieniu będzie zatem akceptowane zarówno stare jak i nowe hasło; w repozytorium Kippo jest dostępny przykładowy plik *userdb.txt*: `http://kippo.googlecode.com/svn/trunk/data/userdb.txt`

- *last.log* - plik tekstowy zawierający informację o ostatnich logowaniach do systemu emulowanego przez Kippo; po zakończeniu każdej uwierzytelnionej sesji SSH Kippo dodaje na końcu tego pliku informację o zakończonej sesji (rejestrowaną nazwą użytkownika jest zawsze root niezależnie od faktycznej nazwy użytkownika, który był zalogowany);

4 Uruchomienie i rekonfiguracja

W celu uruchomienia Kippo należy wykonać skrypt *start.sh*.

Rekonfiguracja honeypota polega na zmodyfikowaniu wyżej omówionych elementów konfiguracyjnych Kippo za pomocą zewnętrznych narzędzi. Zmiany dokonane w głównym pliku konfiguracyjnym, w skrypcie start.sh (parametry wywołania twistd) i/lub w pliku emulującym system plików (tj. zmiany dotyczące elementów konfiguracji omówionych w punktach 3.1-3.3) wymagają restartu honeypota. Natomiast zmiany dotyczące treści plików, komunikatów generowanych przez pliki wykonywalne lub dodatkowych danych konfiguracyjnych (tj. zmiany dotyczące elementów konfiguracji omówionych w punktach 3.4-3.6) są od razu uwzględniane przez działającego honeypota Kippo.

5 Charakterystyka danych wyjściowych

Kippo zbiera i przechowuje dane o różnego rodzaju zdarzeniach, które mają miejsce w ramach poszczególnych połączeń TCP/sesji SSH. Wszystkie te dane są zapisywane w pliku tekstowym pełniącym funkcję głównego logu honeypota. Ponadto przebieg każdej utworzonej sesji SSH jest rejestrowany w oddzielnym pliku binarnym. Opcjonalnie część danych zbieranych przez Kippo może być umieszczana w bazie danych MySQL i/lub przekazywana do serwera XMPP. Należy jednak podkreślić, że w bazie danych i na serwerze XMPP nie są dostępne wszystkie dane wyjściowe, które można znaleźć w głównym logu. Dla każdego połączenia TCP/sesji SSH można znaleźć następujące dane w logu (L), bazie danych (B) i na serwerze XMPP (X) o takich zdarzeniach jak:

- nawiązanie połączenia TCP:
  - źródłowy adres IP: L, B, X;
  - źródłowy port TCP: L, X;
  - docelowy adres IP: L, X;

  - docelowy port TCP: L, X;
  - czas nawiązania połączenia TCP: L, B;
- zakończenie połączenia TCP/sesji SSH:
  - przyczyna zakończenia połączenia TCP/sesji SSH: L;
  - czas zakończenia połączenia TCP/sesji SSH: L, B*, X*;
  - w przypadku zakończenia sesji SSH:
    * zawartość pliku z przebiegiem sesji: B*;
- odebranie informacji o wersji klienta SSH:
  - wersja klienta SSH: L, B*, X*;
  - czas odebrania informacji o wersji klienta: L;
- utworzenie szyfrowanego połączenia:
  - algorytm używany do szyfrowania: L;
  - czas utworzenia szyfrowanego połączenia: L;
- próba uwierzytelnienia klienta:
  - login: L, B, X;
  - hasło: L, B, X;
  - informacja o tym czy próba uwierzytelniania zakończyła się powodzeniem: L, B, X;
  - czas wykonania próby uwierzytelnienia klienta: L, B;
  - w przypadku udanej próby uwierzytelnienia (utworzenia sesji SSH):
    * rozmiar terminala: L, B;
    * nazwa utworzonego pliku z przebiegiem sesji: L;
    * wartości zmiennych środowiskowych (np. XMODIFIERS, LANG): L;
- wykonanie polecenia powłoki w ramach utworzonej sesji SSH:
  - nazwa polecenia: L, B, X;
  - informacja o tym czy polecenie zostało rozpoznane przez Kippo: L, B, X;
  - czas wykonania polecenia: L, B;

  - w przypadku polecenia *wget*:
    * URL pobranego pliku: L, B;
    * nazwa, pod którą w macierzystym systemie został zapisany pobrany plik: L, B;
    * czas rozpoczęcia pobierania pliku: L, B;
    * czas zakończenia pobierania pliku: L;

- wprowadzenie danych wejściowych dla polecenia powłoki (np. *passwd*):
  - nazwa polecenia: L, B;
  - dane wejściowe: L, B;
  - czas wprowadzenia danych wejściowych: L, B;

UWAGA! W przypadku najnowszej, w momencie pisania tej publikacji, wersji Kippo (rewizja 234) część danych nie jest przekazywana do bazy danych (B*) i serwera XMPP (X*), pomimo iż wersja wcześniejsza (rewizja 228 - dla tej wersji z kolei nie działa zapisywanie w bazie danych informacji o pobieranych plikach) te dane rejestrowała w bazie danych i na serwerze XMPP.

Poniżej dokładniej omówiono sposób reprezentacji danych zbieranych i udostępnianych przez Kippo.

6 Format danych wyjściowych

6.1 Główny log Kippo

Kippo zapisuje wszystkie dane o bieżących zdarzeniach w pliku tekstowym o domyślnej nazwie *kippo.log*, którą ewentualnie można zmienić modyfikując parametr wywołania *twistd* w skrypcie *start.sh* (patrz 3.2). Gdy plik *kippo.log* przekroczy rozmiar 1MB, to jego zawartość zostaje przeniesiona do pliku *kippo.log.1*, którego ewentualna poprzednia zawartość (jeżeli ten plik był już wcześniej utworzony w systemie) jest przenoszona do pliku *kippo.log.2*, którego z kolei ewentualna poprzednia zawartość jest przenoszona do pliku *kippo.log.3*,... itd. Na główny log Kippo składa się zatem jeden plik tekstowy, w którym są zapisywane dane o aktualnych zdarzeniach oraz zero lub więcej plików tekstowych z danymi o starszych zdarzeniach.

Wpisy w głównym logu Kippo składają się z trzech, następujących po sobie i rozdzielonych pojedynczymi spacjami części:

- timestamp – czas dodania wpisu;

- context – kontekst, w którym pojawia się wpis;
- message – treść wpisu (może składać się z wielu wierszy);

Czas dodania wpisu jest czasem lokalnym pobranym z systemu i ma następujący format: YYYY-MM-DD HH:mm:ssTZD.

Poniżej omówiono format kontekstu i treści dla wpisu w zależności od zdarzenia, którego wpis dotyczy.

Nawiązanie połączenia TCP

Nawiązanie połączenia TCP jest rejestrowane w logu jako jeden wpis o następującym kontekście i treści:

- context := [kippo.core.honeypot.HoneyPotSSHFactory];
- message := New connection: *IP_source*:*port_source* (*IP_dest*:*port_dest*) [session: *session_id*], gdzie:
  - *IP_source*: źródłowy adres IP;
  - *port_source*: źródłowy port TCP;
  - *IP_dest*: docelowy adres IP;
  - *port_dest*: docelowy port TCP;
  - *session_id*: liczba jednoznacznie identyfikująca połączenie TCP/sesję SSH w ramach logu (identyfikator sesji);

Przykładowy wpis informujący o nawiązaniu połączenia TCP:

```
2013−01−08 14:18:25+0100 [kippo.core.honeypot.HoneyPotSSHFactory]
    New connection: 192.168.122.1:35533 (192.168.122.82:2222) [session: 0]
```

Zakończenie połączenia TCP/sesji SSH

Zakończenie połączenia TCP/sesji SSH jest rejestrowane w logu jako jeden lub dwa wpisy o następującym kontekście:

- context := [HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - *session_id*: identyfikator sesji, która została zakończona;
  - *IP_source*: źródłowy adres IP sesji, która została zakończona;

Wpis oznaczający fakt zakończenia połączenia TCP/sesji SSH ma następującą treść:

- message := connection lost;

Przed tym wpisem może wystąpić wpis informujący o przyczynie zakończenia połączenia TCP/sesji SSH, którego treść w zależności od jednej z dwóch przyczyn jest następująca:

- message := Disconnecting with error, code 2
  reason: bad packet length *length_value*
- message := Got remote error, code 11
  reason: disconnected by user

Obie powyższej treści zajmują dwa wiersze. Pierwsza z powyższych treści odnosi się do przypadku zerwania połączenia z powodu odebrania pakietu o złym rozmiarze *length_value*. Natomiast druga treść oznacza zerwanie połączenia przez klienta.

Przykładowe wpisy informujące o zakończeniu połączenia:

```
2013−01−08 15:03:22+0100 [HoneyPotTransport,0,192.168.122.1] Got
    remote error, code 11
reason: disconnected by user
2013−01−08 15:03:22+0100 [HoneyPotTransport,0,192.168.122.1]
    connection lost
```

Odebranie informacji o wersji klienta SSH

Odebranie informacji o wersji klienta SSH jest rejestrowane w logu jako jeden wpis o następującym kontekście i treści:

- context := [HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - session_id: identyfikator sesji, której dotyczy informacja o wersji klienta SSH;
  - IP_source: źródłowy adres IP sesji, której dotyczy informacja o wersji klienta SSH;
- message := Remote SSH version: *version_string*, gdzie:
  - *version_string*: ciąg znaków opisujący wersję klienta SSH;

Przykładowy wpis informujący o wersji klienta SSH:

```
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] Remote
    SSH version: SSH−2.0−OpenSSH_5.3
```

Utworzenie szyfrowanego połączenia

Utworzenie szyfrowanego połączenia jest rejestrowane w logu jako pięć wpisów o kontekście:

- context := [HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - *session_id*: identyfikator sesji, dla której utworzono szyfrowane połączenie;
  - *IP_source*: źródłowy adres IP sesji, dla której utworzono szyfrowane połączenie;

Z pięciu wpisów dotyczących utworzenia szyfrowanego połączenia najważniejsze są dwa o następujących treściach:

- message := incoming: *cipher_spec*, gdzie:
  - *cipher_spec*: algorytm szyfrowania;
- message := starting service ssh-userauth;

Pierwsza z powyższych treści informuje o algorytmie szyfrowania, jaki będzie używany dla szyfrowanego połączenia. Druga treść stwierdza fakt utworzenia szyfrowanego połączenia.

Przykładowe wpisy informujące o utworzeniu szyfrowanego połączenia:

```
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] kex alg,
    key alg: diffie−hellman−group1−sha1 ssh−rsa
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] outgoing:
     aes128−ctr hmac−md5 none
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] incoming
    : aes128−ctr hmac−md5 none
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] NEW
    KEYS
2013−01−08 14:18:26+0100 [HoneyPotTransport,0,192.168.122.1] starting
    service ssh−userauth
```

Próba uwierzytelnienia klienta

Próba uwierzytelnienia klienta jest rejestrowana w logu jako kilka wpisów o kontekście:

- context := [SSHService ssh-userauth on HoneyPotTransport,*session_id*,*IP_source*], gdzie:

  - *session_id*: identyfikator sesji, której dotyczy próba uwierzytelnienia;
  - *IP_source*: źródłowy adres IP sesji, której dotyczy próba uwierzytelnienia;

Z kilku wpisów dotyczących próby uwierzytelnienia najważniejszy jest wpis o następującej treści:

- message := login attempt [*username*/*password*] *result*, gdzie:
  - username: login użytkownika;
  - password: hasło użytkownika;
  - result: wynik próby uwierzytelnienia (succeeded lub failed);

W przypadku udanej próby uwierzytelnienia (result = succeeded) w logu są umieszczane kolejne ważne wpisy o następującym kontekście:

- context := [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - *session_id*: identyfikator sesji, dla której próba uwierzytelnienia powiodła się;
  - *IP_source*: źródłowy adres IP sesji, dla której próba uwierzytelnienia powiodła się;

Wpisy te zawierają informację o rozmiarze terminala, nazwie utworzonego pliku z przebiegiem sesji i wartościach zmiennych środowiskowych. Odpowiadają im następujące treści:

- message := Terminal size: *row_num col_num*, gdzie:
  - *row_num*: liczba wierszy;
  - *col_num*: liczba kolumn;
- message := Opening TTY log: *file_name*, gdzie:
  - *file_name*: nazwa utworzonego pliku z przebiegiem sesji;
- message := request_env: '\x00\x00\x00\nXMODIFIERS\x00\x00\x00\x08*var_value*', gdzie:
  - *var_value*: wartość zmiennej XMODIFIERS;

- message := request_env: '\x00\x00\x00\x04LANG\x00\x00\x00\n*var_value*', gdzie:
  - *var_value*: wartość zmiennej LANG;

Przykładowe wpisy informujące o próbie uwierzytelnienia:

2013−01−08 14:33:27+0100 [SSHService ssh−userauth on HoneyPotTransport,0,192.168.122.1] login attempt [root/123456] succeeded
2013−01−08 14:33:27+0100 [SSHService ssh−userauth on HoneyPotTransport,0,192.168.122.1] root authenticated with keyboard −interactive
2013−01−08 14:33:27+0100 [SSHService ssh−userauth on HoneyPotTransport,0,192.168.122.1] starting service ssh−connection
2013−01−08 14:33:27+0100 [SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] got channel session request
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] channel open
2013−01−08 14:33:27+0100 [SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] got global no−more−sessions@openssh.com request
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] pty request: xterm (33, 129, 0, 0)
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] Terminal size: 33 129
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] request_env: '\x00\x00\x00\nXMODIFIERS\x00\x00\x00\x08@im=none'
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] request_env: '\x00\x00\x00\x04LANG\x00\x00\x00\npl_PL.utf8'
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] getting shell
2013−01−08 14:33:27+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] Opening TTY log: /var/log/kippo/log/tty/20130108−143327−9152.log
2013−01−08 14:33:33+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] /etc/motd resolved into /etc/motd

Wykonanie polecenia powłoki w ramach utworzonej sesji SSH

Wykonanie polecenia powłoki w ramach utworzonej sesji SSH jest rejestrowane w logu jako dwa wpisy o kontekście:

- context := [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - *session_id*: identyfikator sesji, w ramach której jest wykonywane polecenie powłoki;
  - *IP_source*: źródłowy adres IP sesji, w ramach której jest wykonywane polecenie powłoki;

Z tych dwóch wpisów ważny jest wpis o następującej treści:

- message := *cmd_found* : *cmd_name*, gdzie:
  - *cmd_found*: informacja o tym czy polecenie zostało rozpoznane (Command found) czy nie (Command not found);
  - *cmd_name*: nazwa polecenia powłoki;

W przypadku polecenia *wget* z dalszych dwóch wpisów można uzyskać informację o URL pobieranego pliku, o nazwie, pod którą w macierzystym systemie zostanie zapisany pobrany plik i o czasie rozpoczęcia i zakończenia pobierania pliku. Te wpisy mają następujący kontekst:

- context := [HTTPPageDownloader,client];

Zawierają one odpowiednio następujące treści:

- message := Updating realfile to *file_name*, gdzie:
  - *file_name*: nazwa, pod którą w macierzystym systemie zostanie zapisany pobrany plik;
- message := Stopping factory <HTTPProgressDownloader: *url*>, gdzie:
  - *url*: URL pobieranego pliku;

Przykładowe wpisy informujące o wykonaniu polecenia *wget*:

2013−01−08 14:51:47+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] CMD: wget www.dna.caltech.edu/Papers/DNAorigami−nature.pdf

2013−01−08 14:51:47+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] Command found: wget www.dna.caltech.edu/Papers/DNAorigami−nature.pdf
2013−01−08 14:51:47+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] Starting factory <HTTPProgressDownloader: http://www.dna.caltech.edu/Papers/DNAorigami−nature.pdf>
2013−01−08 14:51:50+0100 [HTTPPageDownloader,client] Updating realfile to /var/log/kippo/dl/20130108145147_http___www_dna_caltech_edu_Papers_DNAorigami_nature_pdf
2013−01−08 14:51:50+0100 [HTTPPageDownloader,client] Stopping factory <HTTPProgressDownloader: http://www.dna.caltech.edu/Papers/DNAorigami−nature.pdf>

Wprowadzenie danych wejściowych dla polecenia powłoki

Wprowadzenie danych wejściowych dla polecenia powłoki jest w logu rejestrowane jako jeden wpis o następującym kontekście i treści:

- context := [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,*session_id*,*IP_source*], gdzie:
  - *session_id*: identyfikator sesji, w ramach której jest wykonywane polecenie powłoki;
  - *IP_source*: źródłowy adres IP sesji, w ramach której jest wykonywane polecenie powłoki;
- message := INPUT (*cmd*): *input_value*, gdzie:
  - *cmd*: nazwa polecenia powłoki;
  - *input_value*: dane wejściowe;

Przykładowy wpis informujący o wprowadzeniu danych dla polecenia powłoki:

2013−01−08 14:51:47+0100 [SSHChannel session (0) on SSHService ssh−connection on HoneyPotTransport,0,192.168.122.1] INPUT (passwd): pp

6.2 Baza danych MySQL

Kippo umożliwia składowanie części zbieranych danych w relacyjnej bazie danych MySQL. Rysunek 1 przedstawia schemat bazy danych honeypota.

W bazie jest zdefiniowanych siedem tabel:

- clients - tabela zawierająca informacje o wersjach klientów SSH; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - version: wersja klienta SSH;
- sensors - tabela zawierająca informacje o instancjach Kippo; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - ip: nazwa identyfikująca instancję Kippo ustalona w głównym pliku konfiguracyjnym (parametr sensor_name);
- sessions - tabela zawierająca informacje o połączeniach TCP/sesjach SSH; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - starttime: czas rozpoczęcia połączenia TCP.sesji SSH;
  - endtime: czas zakończenia połączenia TCP.sesji SSH;
  - sensor: UID instancji Kippo obsługującej połączenie TCP/sesję SSH (FK do sensors);
  - ip: źródłowy adres IP;
  - termsize: rozmiar terminala;
  - client: UID wersji klienta SSH (FK do clients);
- auth - tabela zawierająca informacje o próbach uwierzytelnienia; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - session: UID sesji (FK do sessions);
  - success: informacja o tym czy próba uwierzytelnienia jest udana (1) czy nie (0);
  - username: login użytkownika;
  - password: hasło użytkownika;

  - timestamp: czas przeprowadzenia próby uwierzytelnienia;

- input - tabela zawierająca informacje o wykonanych poleceniach powłoki i o wprowadzonych dla nich danych wejściowych; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - session: UID sesji (FK do sessions);
  - timestamp: czas wykonania polecenia powłoki/wprowadzenia danych wejściowych;
  - realm: w przypadku wykonania polecenia zawsze wartość NULL, a w przypadku wprowadzenia danych wejściowych nazwa polecenia powłoki;
  - success: w przypadku wykonania polecenia informacja o tym czy polecenie zostało przez Kippo rozpoznane (1) czy nie (0), a w przypadku wprowadzenia danych wejściowych zawsze wartość NULL;
  - input: nazwa polecenia powłoki/dane wejściowe;

- downloads - tabela zawierająca informacje o pobranych plikach; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - session: UID sesji (FK do sessions);
  - timestamp: czas rozpoczęcia pobierania pliku;
  - url: URL pobranego pliku;
  - outfile: nazwa, pod którą w macierzystym systemie został zapisany pobrany plik;

- ttylog - tabela zawierająca zawartości plików z przebiegami poszczególnych sesji SSH; ; ta tabela ma następujące kolumny:
  - id: sztuczny UID;
  - session: UID sesji (FK do sessions);
  - ttylog: zawartość pliku z przebiegiem sesji SSH;

UWAGA! Czas w bazie danych jest zapisywany jako UTC.

Poniżej na przykładowych danych omówiono operacje, które są wykonywane na bazie danych w przypadku poszczególnych rodzajów zdarzeń.

Nawiązanie połączenia TCP

W przypadku gdy dana instancja Kippo nie wprowadziła jeszcze do bazy danych żadnej informacji o jakimś połączeniu TCP, to informacja o tej instancji jest umieszczana w tabeli sensors:

```
insert into sensors (id, ip) values ('3', 'kippo_hp');
```

Informacja o nowo nawiązanym połączeniu jest rejestrowana w tabeli sessions:

```
insert into sessions (id, starttime, endtime, sensor, ip, termsize, client)
values ('e28678b4599511e2bab10800277e980c', '2013-01-08 13:18:26',
    NULL, '3', '192.168.122.1', NULL, NULL);
```

Zakończenie połączenia TCP/sesji SSH

Informacja o czasie zakończenia połączenia TCP/sesji SSH jest umieszczana w odpowiednim wierszu tabeli sessions:

```
update sessions set endtime = '2013-01-08 14:03:22' where id = '
    e28678b4599511e2bab10800277e980c';
```

W przypadku zakończenia sesji SSH w tabeli ttylog jest umieszczana zawartość pliku z przebiegiem sesji:

```
insert into ttylog (id, session, ttylog) values ('4', '
    e28678b4599511e2bab10800277e980c', BLOB);
```

Odebranie informacji o wersji klienta SSH

W przypadku gdy w bazie danych nie ma informacji o odebranej wersji klienta, to jest ona wprowadzana do tabeli clients:

```
insert into clients (id, version) values ('2', 'SSH-2.0-OpenSSH_5.3');
```

Informacja o wersji klienta jest kojarzona z reprezentującym odpowiednią sesję wierszem tabeli sessions:

```
update sessions set client = 3 where id = '
    e28678b4599511e2bab10800277e980c';
```

Próba uwierzytelnienia klienta

Informacja o próbie uwierzytelnienia klienta jest umieszczana w tabeli auth:

```
insert into auth (id, session, success, username, password, timestamp)
values ('12', 'e28678b4599511e2bab10800277e980c', '1', 'root', '123456',
    '2013-01-08 13:33:27');
```

W przypadku udanej próby uwierzytelnienia informacja o rozmiarze terminala jest umieszczana w odpowiednim wierszu tabeli sessions:

```
update sessions set termsize = '129x33' where id = '
    e28678b4599511e2bab10800277e980c';
```

Wykonanie polecenia powłoki w ramach utworzonej sesji SSH

Informacja o wykonaniu polecenia powłoki jest umieszczana w tabeli input:

```
insert into input (id, session, timestamp, realm, success, input)
values ('27', 'e28678b4599511e2bab10800277e980c', '2013-01-08 13:42:03',
     NULL, '1', 'pwd')
```

W przypadku polecenia *wget* informacja o pobranym pliku jest umieszczana w tabeli downloads:

```
insert downloads (id, session, timestamp, url, outfile)
values ('1', 'e28678b4599511e2bab10800277e980c', '2013-01-08 13:42:03', '
    http://cachefly.cachefly.net/100mb.test', '/var/log/kippo/dl
    /20130131032708_http___cachefly_cachefly_net_100mb_test' );
```

Wprowadzenie danych wejściowych dla polecenia powłoki

Informacja o wprowadzeniu danych dla polecenia powłoki jest umieszczana w tabeli input:

```
insert into input (id, session, timestamp, realm, success, input)
values ('27', 'e28678b4599511e2bab10800277e980c', '2013-01-08 13:42:03',
     'passwd', NULL, 'pp');
```

6.3 Serwer XMPP

Kippo może przesyłać część zbieranych danych do serwera XMPP, za pośrednictwem którego można na bieżąco odbierać wiadomości XMPP informujące o:

- utworzeniu połączenia TCP/sesji SSH;
- zakończeniu połączenia TCP/sesji SSH;
- wersji klienta SSH;
- nieudanej próbie uwierzytelnienia;
- udanej próbie uwierzytelnienia;
- wykonaniu polecenia powłoki;

Z każdym z powyższych rodzajów wiadomości XMPP może być związany odrębny pokój MUC, do którego będą przesyłane te wiadomości zgodnie z protokołem XMPP.

Poniżej podano przykładowe wiadomości XMPP.

Wiadomość informująca o utworzeniu połączenia TCP/sesji SSH

```
<message from="kippo-events-createsession@conference.localhost/kippo
    -XDJQcVxo" type="groupchat" to="kkoltys@localhost/localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="createsession">
<session local_host="127.0.0.1" session="136371692
    cdb4d91b3eec6ff04618232" local_port="2222" remote_port="35533"
    remote_host="192.168.122.1"/>
</kippo>
</body>
</message>
```

Wiadomość informująca o zakończeniu połączenia TCP/sesji SSH

```
<message from="kippo-events-connectionlost@conference.localhost/
    kippo-XDJQcVxo" type="groupchat" to="kkoltys@localhost/
    localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="connectionlost
    ">
<session session="136371692cdb4d91b3eec6ff04618232"/>
</kippo>
</body>
</message>
```

Wiadomość informująca o wersji klienta SSH

```
<message from="kippo-events-clientversion@conference.localhost/kippo
    -XDJQcVxo" type="groupchat" to="kkoltys@localhost/localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="clientversion">
<version session="136371692cdb4d91b3eec6ff04618232" version="SSH
    -2.0-OpenSSH_5.3"/>
</kippo>
</body>
</message>
```

Wiadomość informująca o nieudanej próbie uwierzytelnienia

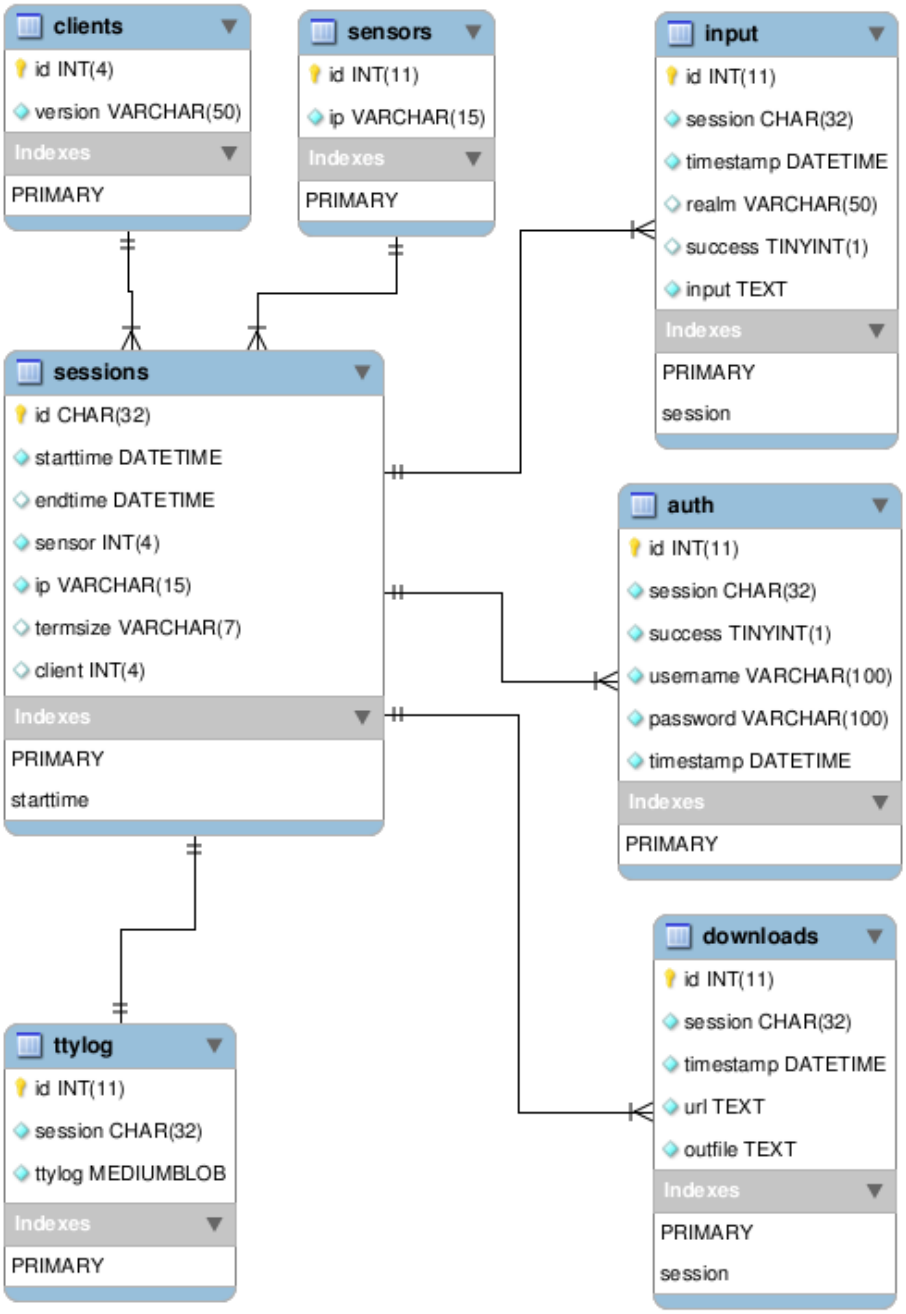
```
<message from="kippo-events-loginfailed@conference.localhost/kippo-
    XDJQcVxo" type="groupchat" to="kkoltys@localhost/localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="loginfailed">
<credentials session="136371692cdb4d91b3eec6ff04618232" password
    ="123" username="root"/>
</kippo>
</body>
</message>
```

Wiadomość informująca o udanej próbie uwierzytelnienia

```
<message from="kippo-events-loginsucceded@conference.localhost/kippo
    -XDJQcVxo" type="groupchat" to="kkoltys@localhost/localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="loginsucceeded
    ">
<credentials session="136371692cdb4d91b3eec6ff04618232" password
    ="123456" username="root"/>
</kippo>
</body>
</message>
```

Wiadomość informująca o wykonaniu polecenia powłoki

```
<message from="kippo-events-command@conference.localhost/kippo-
    XDJQcVxo" type="groupchat" to="kkoltys@localhost/localhost">
<body>
<kippo xmlns="http://code.google.com/p/kippo/" type="command">
<command session="136371692cdb4d91b3eec6ff04618232" command="
    known">pwd</command>
</kippo>
</body>
</message>
```



Rysunek 1: Schemat bazy danych honeypota Kippo