



# Generowanie szelkodów w Pythonie

Piotr Sobolewski



**Bywa, że potrzebny jest nam nietypowy szelkod. Bywa, że szelkod musi być tworzony dynamicznie, podczas wykonywania się eksploita. W takiej sytuacji przydać się może *InlineEgg* – biblioteka umożliwiająca pisanie w Pythonie programów generujących szelkody.**

**D**o stworzenia eksploita wykorzystującego błąd przepełnienia bufora potrzebny jest szelkod – mały programik, napisany w asemblerze, który zostanie przesłany do dziurawego programu i wykonany w atakowanym systemie. Szelkod jest zwykle zaszyty na sztywno w eksploicie. Ponieważ czynności, które ma wykonać, są zazwyczaj standardowe (zmiana UID na zero, uruchomienie powłoki itp.), więc autor eksploita nie musi pisać szelkodu samemu, może skorzystać z gotowego, na przykład znalezionego w Internecie.

Bywają sytuacje, kiedy takie podejście nie wystarcza. Zdarza się, że potrzebny jest nam nietypowy szelkod, który zrobi coś, czego nie robi żaden z powszechnie dostępnych. Bywa, że szelkod musi być tworzony dynamicznie, podczas wykonywania się eksploita, dostosowany do konkretnej sytuacji. W takiej sytuacji przydać się może *InlineEgg* – biblioteka umożliwiająca pisanie w Pythonie programów generujących szelkody.

## Koncepcja

Koncepcja korzystania z *InlineEgg* jest prosta (patrz Rysunek 1): aby wygenerować szelkod

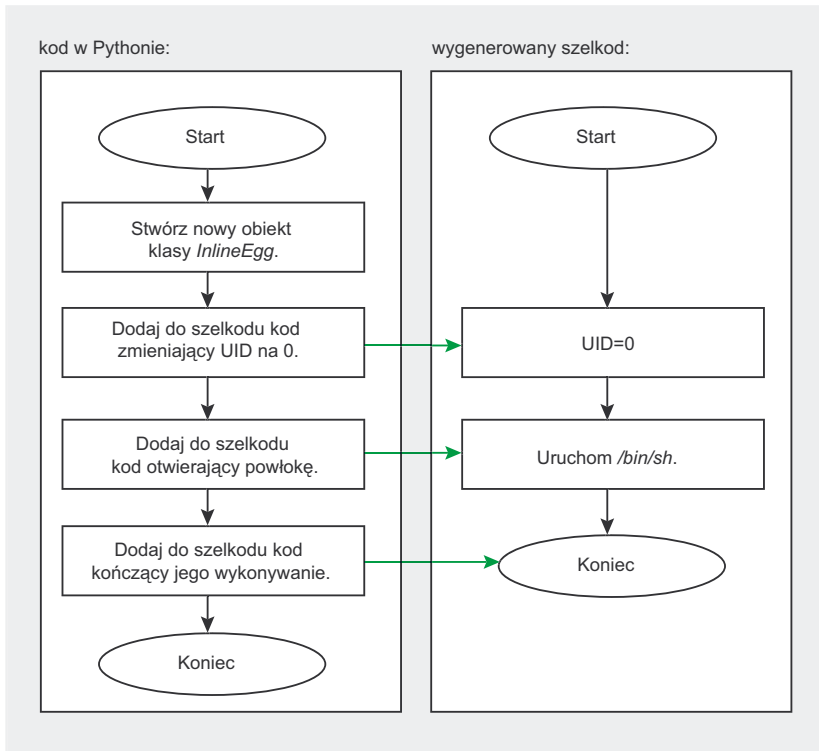
tworzymy obiekt klasy *InlineEgg*, a następnie wywołujemy odpowiednie metody, na przykład *dodaj do szelkodu kod zmieniający uid na zero*, *dodaj do szelkodu kod uruchamiający*

## Z artykułu nauczysz się...

- będziesz umiał napisać własny exploit przy użyciu biblioteki *InlineEgg*.

## Powinieneś wiedzieć...

- Zakładamy, że Czytelnik rozumie, na czym polega przepełnianie bufora na stosie (na przykład przeczytał Artykuł *Przepełnianie stosu pod Linuxem x86* w poprzednim numerze naszego pisma),
- Opisujemy skrótowo, na czym polega programowanie gniazd sieciowych; warto uzupełnić wiedzę na przykład czytając dołączony na CD (w dziale *dokumentacja->programowanie->C*) dokument *Beej's Guide to Network Programming*,
- Do pisania szelkodów wykorzystujemy Pythona, jednak nie zakładamy, że Czytelnik ma doświadczenie z tym językiem, powinno wystarczyć ogólne doświadczenie z programowaniem.



Rysunek 1. Generowanie szelkodu za pomocą *InlineEgg*

powłokę. Każda z wywołanych metod dodaje do tworzonego szelkodu kawałek kodu w języku maszynowym wybranego procesora. Na koniec otrzymujemy gotowy do użycia szelkod.

## Nasz pierwszy szelkod

Spróbujmy napisać własny szelkod za pomocą *InlineEgg* (Listing 1). Zaczniemy od czegoś prostego: stwórzmy szelkod, który wykonuje polecenie `ls` (wypisuje zawartość bieżącego katalogu).

Zaczynamy od stworzenia klasy reprezentującej szelkod:

```
egg=InlineEgg(Linuxx86Syscall)
```

Użyty argument `Linuxx86Syscall` oznacza platformę, dla której generujemy szelkod: w tym przypadku jest to Linux na x86.

Następnie musimy dodać do szelkodu kod wykonujący polecenie `ls`. Odbywa się to przez wywołanie metod klasy *InlineEgg* (patrz Tabela 1). Po przejrzaniu tabeli zauważymy, że do naszego celu znakomicie nada się metoda `execve()`, uruchamiająca dowolny program.

Jako argument podajemy jej ścieżkę do programu, który chcemy uruchomić (czyli `ls`), i argumenty, z jakimi uruchomiony zostanie program. Sprawdźmy ścieżkę do programu `ls`:

```
$ which ls
/bin/ls
```

Podając argumenty nie zapomnijmy, że każdy program uruchamiany pod Linuxem oprócz zwykłych argumentów otrzymuje jako argument pierwszy własną nazwę (patrz Ramka *Pierwszy argument – nazwa programu*). Pierwszym argumentem będzie więc ciąg `ls`. Jako drugi argument podajmy na przykład `-l`, co spowoduje, że wypisane zostaną dodatkowe informacje o plikach.

Ostatecznie linijka powodująca dodanie do tworzonego szelkodu kawałka kodu, który uruchamia polecenie `ls -l`, wygląda tak:

```
egg.execve('/bin/ls', ('ls', '-l'))
```

Ostatnim elementem szelkodu będzie kod kończący jego działanie.

Listing 1. *egg\_1.py* – najprostszy szelkod stworzony za pomocą *InlineEgg*

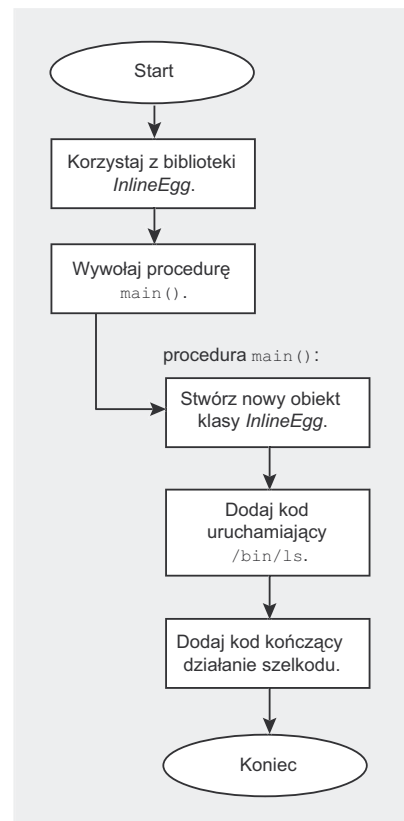
```
#!/usr/bin/python

from inlineegg import *

def main():
    egg = InlineEgg(Linuxx86Syscall)
    egg.execve('/bin/ls', ('ls', '-l'))
    egg.exit()

print egg.getCode()

main()
```



Rysunek 2. Schemat działania programu z Listingu 1

Metoda dodająca taki kod (patrz Tabela 1) to `exit()`:

```
egg.exit()
```

Spróbujmy, czy rzeczywiście w ten sposób da się stworzyć szelkod. Przeanalizujemy program przedstawiony na Listingu 1. Z jego zrozumieniem nie powinny mieć kłopotu nawet osoby, które pierwszy raz styka-