МЕХАНИЗМЫ ЗАЩИТЫ ИСХОДНОГО КОДА ИНТЕРПРЕТИРУЕМЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ В СИСТЕМАХ ТЕСТИРОВАНИЯ УЧЕБНЫХ ПРОГРАММ

А.М. Кадан, А.Б. Перовский
Кафедра системного программирования и компьютерной безопасности,
Гродненский государственный университет им.Я.Купалы
Гродно, Республика Беларусь
E-mail: kadan@mf.grsu.by, perovskijab@gmail.com

Представлены методология использования и компоненты системы тестового контроля знаний, используемой студентами, специализирующимися в области ИТ и программирования. К особенностям предлагаемой системы относятся использование интерпретируемого языка программирования Python и требование обеспечения защиты исходного кода от несанционированного изучения.

Введение

По мере расширения сферы применения информационных систем и технологий возникают новые задачи, связанные с использованием нетрадиционных для их предметной области решений. Такие решения позволяют, в частности, добиться нового качества при решении известных задач, но требуют соблюдения определенных соглашений по их использованию.

В настоящее время, в связи с позитивным восприятием субъектами национальной системы образования прогрессивных идей зарубежных академических школ, начал активно проявляться практический интерес к использованию программных средств их сопровождения. В частности, в подготовке студентов ИТ-специальностей это проявляется в активном переходе к дистационным формам контроля, уменьшению доли использования на начальном этапе обучения профессиональных систем программирования и в использовании интерпретируемых языков программирования типа Python, PHP, Javascript и др.

I. Обзор систем удаленного тестирования

В данной работе нас интересует обучение интерпретируемым языкам программирования, в частности – Python. Язык Python весьма популярен в среде американских университетов. Python входит в Top10 самых популярных языков программирования мира [1].

Среди систем и сервисов, которые предоставляют возможность тестирования исходного кода на интерактивных языках программирования, наиболее известные – Coursera [2], Code Academy [3], LearnStreet [4].

В каждой из таких систем есть одно ключевое условие – для каждого задания сформирован фиксированный список имен тестируемых переменных (функций), что позволяет точно знать, какой тест необходимо провести над значениями этих переменных и операторами их вычисления. Если имя переменной не соответствует ожидаемому, система не проверяет результат работы и сообщает об ошибке.

Большинство подобных систем представляют собой клиент-серверные приложения, где серверная компонента выполняет основную работу, оставляя клиентской компоненте функции интерфейса.

II. Дополнительные требования к заданиям

Во-первых, для нас представляют интерес ситуации, которые предполагают не только контроль соответствия заранее формированного тестового файла результату, сформированному программой. Использование интерпретируемого языка программирования допускает анализ исходного кода разработанной программы. Поэтому важен не только «числовой» ответ, но и правильность рассуждений, позволяющих его получить. Причем использование испытуемым конструкций языка программирования может быть заранее регламентировано.

К примеру, решение, с использованием языка Python, задачи нахождения энтропии:

```
# Вычислить энтропию Д.С.В.,
# представленной вектором вероятностей.
# Результат - в битах/символ
P = {'x0':1/2, 'x1':1/4, 'x2':1/8, 'x3':1/8}
H = ...
```

может быть выполнено несколькими принципиально различными способами, использующими различные вычислительные аспекты языка Python и различный формат представления результата. От:

```
P = {'x0':1/2, 'x1':1/4, 'x2':1/8, 'x3':1/8}
H = 1.75
```

до

```
from math import log
P = {'x0':1/2, 'x1':1/4, 'x2':1/8, 'x3':1/8}
def entro(a):
   return sum(-a[x]*log(a[x],2) for x in a)
H = entro(P)
```

Во-вторых, важным аспектом является то, что наряду с проверкой правильности отдельных программ нас интересует работа с «рабочими листами», содержащими целые наборы взаимосвязанных заданий, зачастую использующих модули из предварительно разработанных внешних библиотек.

В этом случае традиционные схемы удаленной проверки отдельных заданий не применимы. В частности, необходима пересылка серверной компоненте нескольких файлов для проверки.

В-третьих, эффективным представляется предварительное тестирование разрабатываемых программ с использованием заранее подготовленной системы тестов, что существенно повысило бы эффективность работы, так как навыки самостоятельного тестирования неэлементарных проиложений требуют определенного опыта.

III. Угрозы безопасности тестирующей системы для интерпретируемых языков

Важная особенность пердлагаемой системы состоит в том, что все ее модули будут находится на клиентской машине. Модули, которые были на серверной стороне, будут выполнятся у клиента, либо вовсе отсутствовать. Ресурсоёмкость самостоятельного приложения станет выше, так как все операции выполняются на одной машине.

Обычно такие приложения имеют повышенный уровень защиты интеллектуальной собственности, содержащейся в исходном коде приложения. Причина заключается в том, что все программные модули находятся в полном распоряжении пользователей и подвержены повышенному риску анализа. Еще большему риску подвергнуты приложения, написанные на интерпретируемых языках программирования, т.к. их исходный код полностью открыт, если к нему не применены специальные техники защиты исходного кода.

Данная работа связана с разработкой приложений на языке Python, который является интерпретируемым. Поэтому исходный код такого приложения вместе со всеми модулями будет доступен для исследования. В связи с этим следует позаботиться о сокрытии наиболее критических частей системы: модулей тестирования, в которых хранятся тестовые значения и эталонные решения задач.

IV. Механизмы защиты исходного кода

Очевидный недостаток интерпретируемых языков программирования – исходный код приложения открыт, в отличии от компилируемых языков программирования. Если в модуле, в котором осуществляется проверка решения, помещены эталонные функции или тесты, любой студент сможет воспользоваться этими эталонами для решения своего задания. В связи с этим необходимо выполнять сокрытие исходного кода от глаз пользователей такой системы. Для этого существует несколько способов, основные из них:

- обфускация – запутывание кода (удаление комментариев и строк документации; изменение отступов; добавление пробелов между лексемами; переименование классов, функций и переменных);
- замена опкодов компилятора для того, чтобы декомпиляторы не могли извлечь исходный код из байт-кода;
- компилирование исходных кодов в библиотечные файлы (с расширением pyd);
- шифрование – наиболее мощный и трудно поддающийся реверсивному инжинирингу способ. Трудность в реализации состоит в том, что необходимо скрытым образом расшифровывать импортируемые модули;
- комбинированные способы.

V. Элементы архитектуры системы

Основываясь на предложенных требованиях, можно выделить следующие архитектурные компоненты такой тестирующей системы. Она включает три основных модуля и один вспомогательный, предназначенный для защиты исходного кода клиентской компоненты. Первый модуль (Exercise's Module) содержит решения задач, предложенные пользователем. Второй модуль (Client Module) выполняет проверку решений и запускает третий модуль (Test Module), в котором хранятся эталонные решения заданий и специально подобранные тесты, которые будут применяться к предлагаемым решениям.

Заключение

К настоящему времени спроектирован и частично реализован прототип представленной системы. Прототип функционирует в рамках корпоративной образовательной среды кафедры системного программирования и компьютерной безопасности Гродненского госуниверситета и используется при чтении дисциплин «Вычислительная практика» и «Теория информации» студентам ряда ИТ-специальностей. Весьма активно изучаются вопросы безопасности его использования и устойчивости к атакам.

1. TIOBE Index for September 2014 / TOIBE Software [Электронный ресурс]. – Режим доступа: http://www.tiobe.com/index.php/content/paperinfo/tpci/. – Дата доступа: 13.09.2014.
2. Coursera — проект в сфере онлайн-образования [Электронный ресурс]. – Режим доступа: https://www.coursera.org/. – Дата доступа: 13.09.2014.
3. Code Academy – интерактивное обучение языкам программирования и разметке [Электронный ресурс]. – Режим доступа: http://www.codecademy.com/. – Дата доступа: 13.09.2014.
4. LearnStreet – интерактивное обучение языкам программирования [Электронный ресурс]. – Режим доступа: https://www.learnstreet.com/. – Дата доступа: 23.05.2014.