

КЛАССИФИКАЦИЯ ROP ГАДЖЕТОВ

Алексей Вишняков
vishnya@ispras.ru

Москва, 02 декабря 2016 г.

Актуальность

- В современных программах могут присутствовать тысячи программных дефектов
- Техника возвратно-ориентированного программирования (ROP) может быть применена для использования программных дефектов в злонамеренных целях в условиях работы современных защитных механизмов
- Важно оценить возможность применения ROP

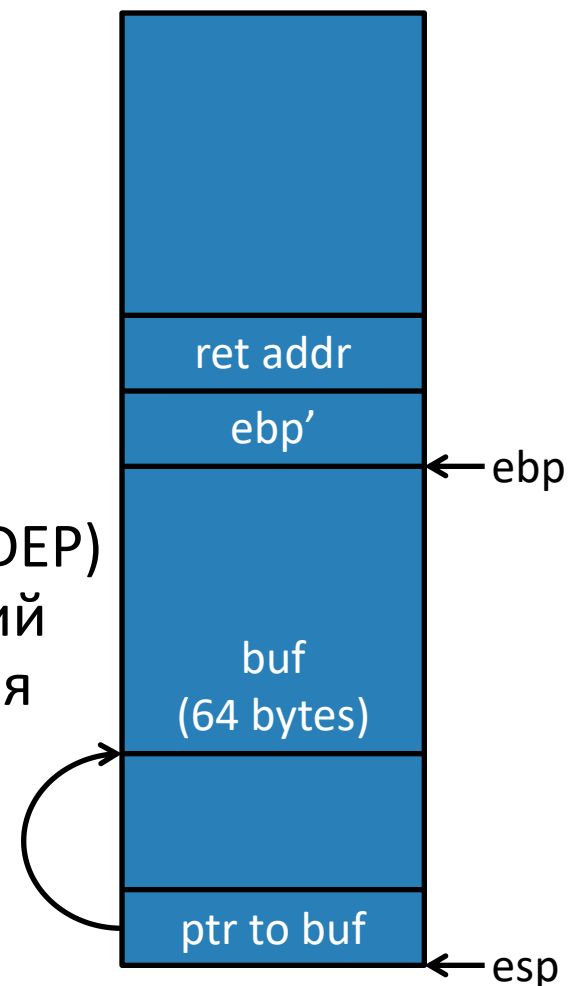
Уязвимость переполнения буфера на стеке

Выполнение вредоносного кода:

- Код размещается на стеке
- Адрес возврата перезаписывается указателем на этот код

Защитный механизм:

- Предотвращение выполнения данных (DEP) – механизм защиты памяти, помечающий страницы недоступными для исполнения (NX бит)
- Стек недоступен для исполнения



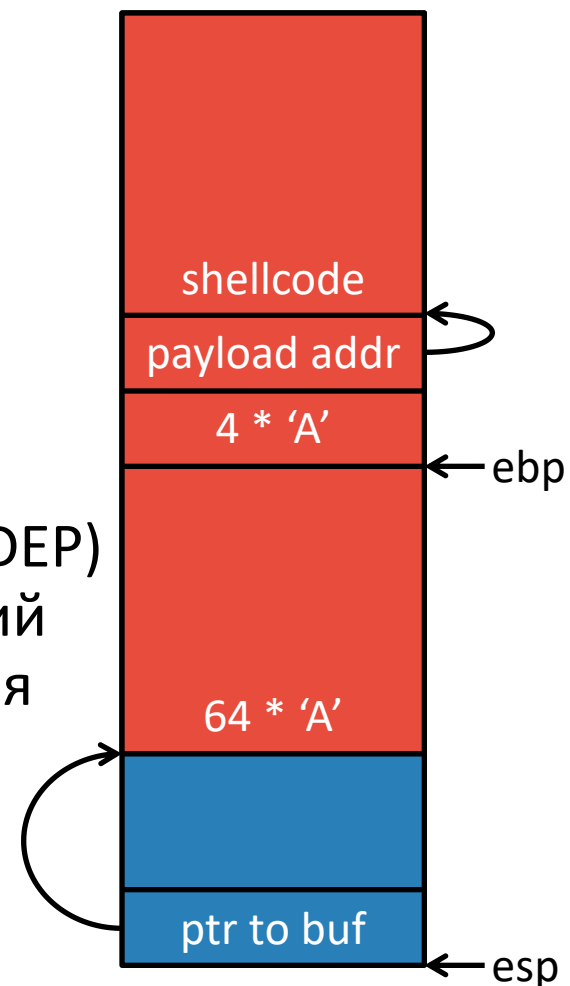
Уязвимость переполнения буфера на стеке

Выполнение вредоносного кода:

- Код размещается на стеке
- Адрес возврата перезаписывается указателем на этот код

Защитный механизм:

- Предотвращение выполнения данных (DEP) – механизм защиты памяти, помечающий страницы недоступными для исполнения (NX бит)
- Стек недоступен для исполнения



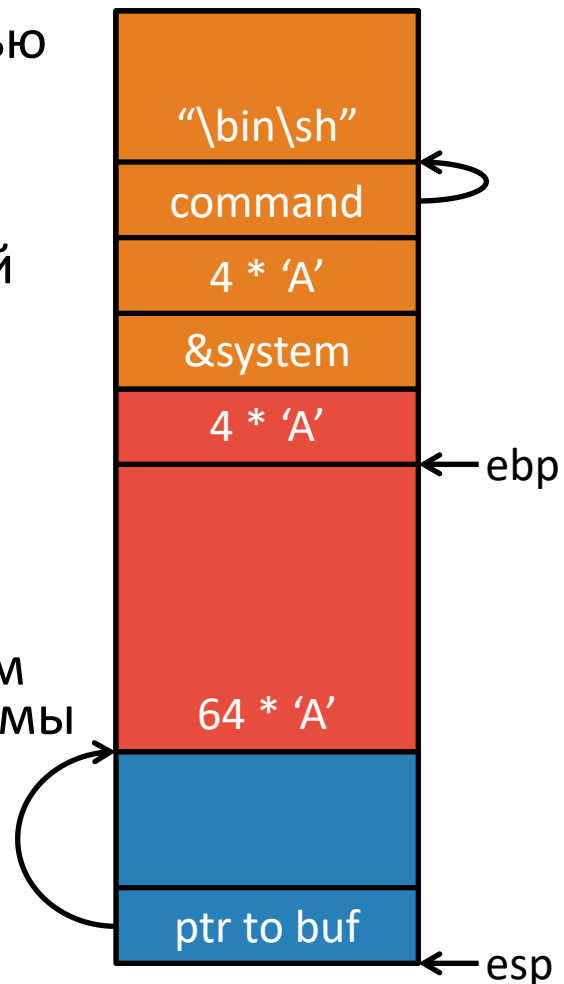
Атака возврата в библиотеку

Выполнение произвольной команды с помощью библиотечной функции:

- Адрес возврата подменяется адресом библиотечной функции
- Выше на стеке размещаются аргументы этой функции
- Позволяет обойти DEP

Защитный механизм:

- Механизм рандомизации адресного пространства (ASLR) позволяет разместить важные структуры программы по различным адресам во время каждого запуска программы
- Адрес библиотечной функции рандомизирован



Возвратно-ориентированное программирование (ROP)

- В Linux адрес загрузки большинства исполняемых файлов остается нерандомизированным
- Гаджет – последовательность инструкций, заканчивающаяся инструкцией передачи управления (`add eax, ebx ; ret`) из нерандомизированных исполняемых секций программы
- Возвратно-ориентированное программирование (ROP) – метод, позволяющий сформировать вредоносный код в виде цепочки гаджетов (ROP-цепочки): первый гаджет передает управление второму, второй – третьему и т.д.

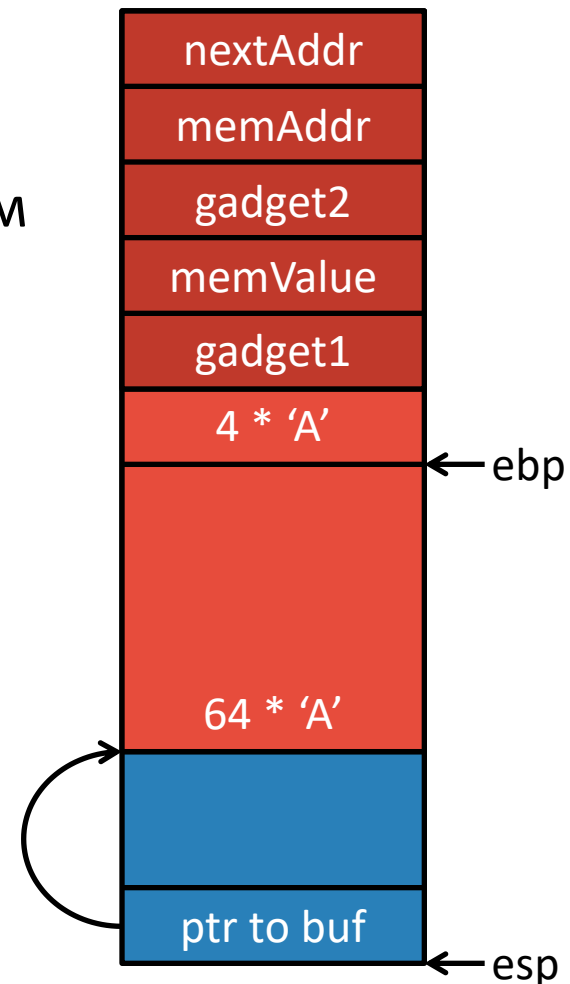
Пример ROP-цепочки

Запись произвольного значения по произвольному адресу:

- Адрес возврата переписывается адресом первого гаджета `gadget1`
- После возврата из первого гаджета управление передается на `gadget2`
- Параметры гаджетов размещаются на стеке

```
gadget1: pop eax  
         ret
```

```
gadget2: pop ebp  
         mov [ebp], eax  
         ret
```



Постановка задачи

- Для оценки качества реализованных защитных механизмов (ИСП ОБФУСКАТОР) требуется уметь определять трудоемкость атаки
- Для формирования нетривиальной ROP-цепочки необходимо наличие достаточного набора гаджетов
- Важно оценить потенциальную возможность формирования такой ROP-цепочки из гаджетов, имеющихся в программе
 - Поиск гаджетов
 - Классификация гаджетов: выделение гаджетов, пригодных для составления ROP-цепочек
 - Оценка пригодности полученного набора гаджетов

Поиск гаджетов

- Алгоритм Галилео
 - Поиск инструкций передачи управления в исполняемых секциях программы
 - Для каждой найденной инструкции производится дизассемблирование нескольких байт, предшествующих инструкции
 - Все корректно дизассемблированные последовательности инструкций добавляются в набор найденных гаджетов
- Алгоритм реализован в ROPgadget

Классификация гаджетов

- Для определения возможности использования найденных гаджетов необходимо иметь представление об их семантике
- Семантика гаджетов может быть определена с помощью их классификации по определенным типам
- За основу взята классификация, использованная в инструменте Q (Edward J. Schwartz, Thanassis Avgerinos, David Brumley, Carnegie Mellon University)
- Классификация расширена дополнительными типами

Типы гаджетов

Тип	Семантика	Пример
NoOpG	Не меняет ничего в памяти и на регистрах	ret
JumpG	$IP \leftarrow \text{AddrReg}$	jmp eax
MoveRegG	$\text{OutReg} \leftarrow \text{InReg}$	mov eax, ebx ; ret
LoadConstG	$\text{OutReg} \leftarrow [\text{SP} + \text{Offset}]$	pop eax ; ret
ArithmeticG	$\text{OutReg} \leftarrow \text{InReg1} \circ \text{InReg2}$	add eax, ebx ; ret
LoadMemG	$\text{OutReg} \leftarrow [\text{AddrReg} + \text{Offset}]$	mov eax, [ebx+0x123] ; ret
StoreMemG	$[\text{AddrReg} + \text{Offset}] \leftarrow \text{InReg}$	mov [eax-0x456], ebx ; ret
ArithmeticLoadG	$\text{OutReg} \circ \leftarrow [\text{AddrReg} + \text{Offset}]$	add eax, [ebx+0x123] ; ret
ArithmeticStoreG	$[\text{AddrReg} + \text{Offset}] \circ \leftarrow \text{InReg}$	add [eax-0x456], ebx ; ret
InitConstG	$\text{OutReg} \leftarrow \text{Value}$	xor eax, eax ; ret
ShiftStackG	$\text{SP} \circ \leftarrow \text{Offset}$	sub esp, 8 ; ret
ArithmeticStackG	$\text{SP} \circ \leftarrow \text{InReg}$	add esp, eax ; ret

Типы гаджетов

- Загрузка константы (LoadConstG)
- Операции с регистрами (MoveRegG, ArithmeticG, ...)
- Операции с памятью (LoadMemG, StoreMemG, ...)
- Передача управления (JumpG)
- Гаджеты-трамплины (ShiftStackG, ArithmeticStackG)
 - Смещают указатель стека
 - Могут быть использованы для обхода «канарейки», если в программе присутствует уязвимость переполнения буфера на стеке при условии «write-what-where»

Классификация гаджетов

Результаты классификации гаджета:

- Типы и параметры гаджета
- Список «испорченных» регистров (значения которых не сохраняются в результате выполнения гаджета)
- Информация о фрейме гаджета (размер фрейма, смещение ячейки с адресом следующего гаджета)

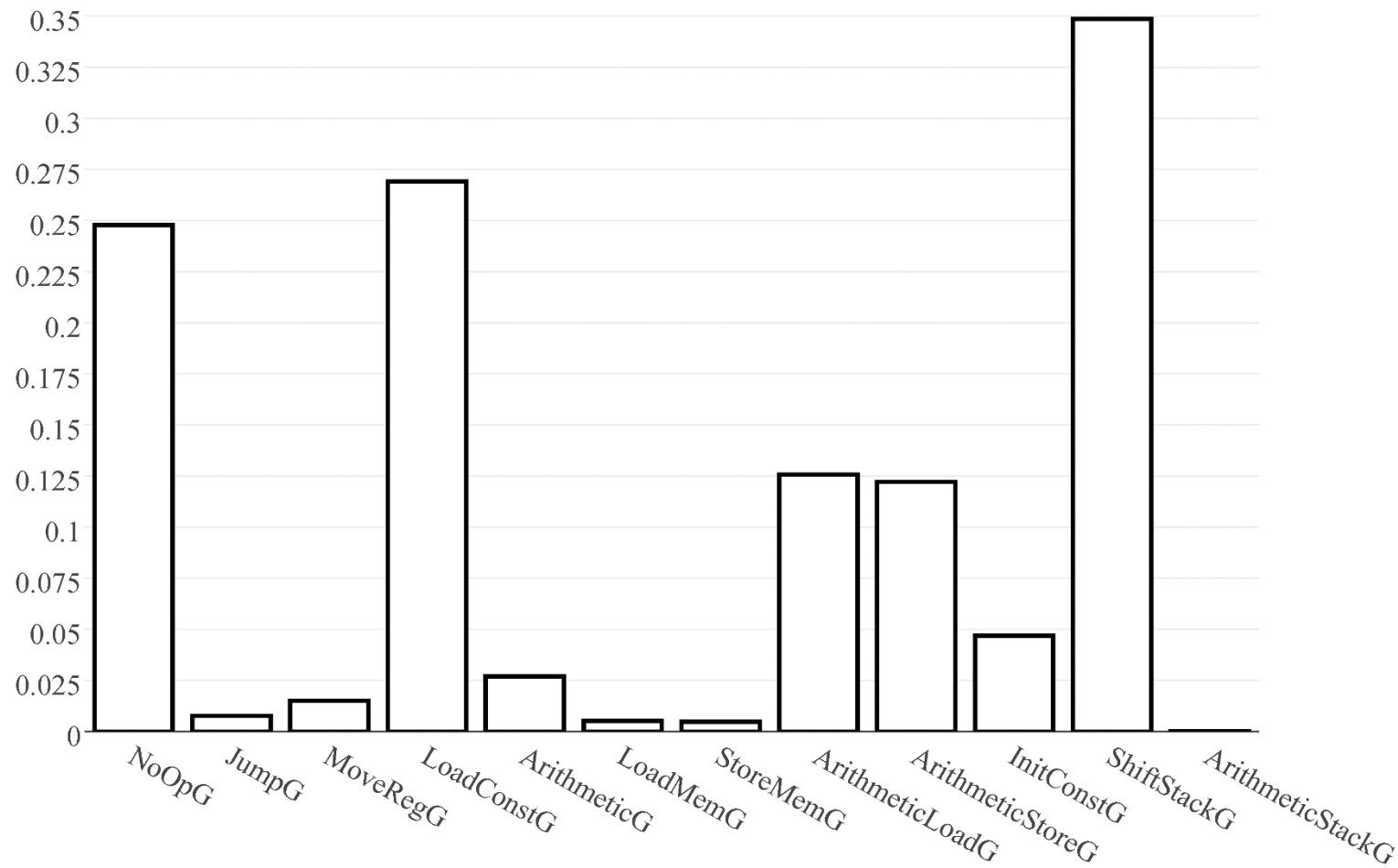
Метод классификации

- Классификация производится на основе анализа эффектов выполнения гаджета на конкретных входных данных
- Инструкции гаджета транслируются в промежуточное представление
- Запускается процесс интерпретации промежуточного представления
 - Отслеживаются обращения к регистрам и памяти
 - Начальные значения регистров и областей памяти генерируются случайным образом
 - Результатом интерпретации будут начальные и конечные значения регистров и памяти

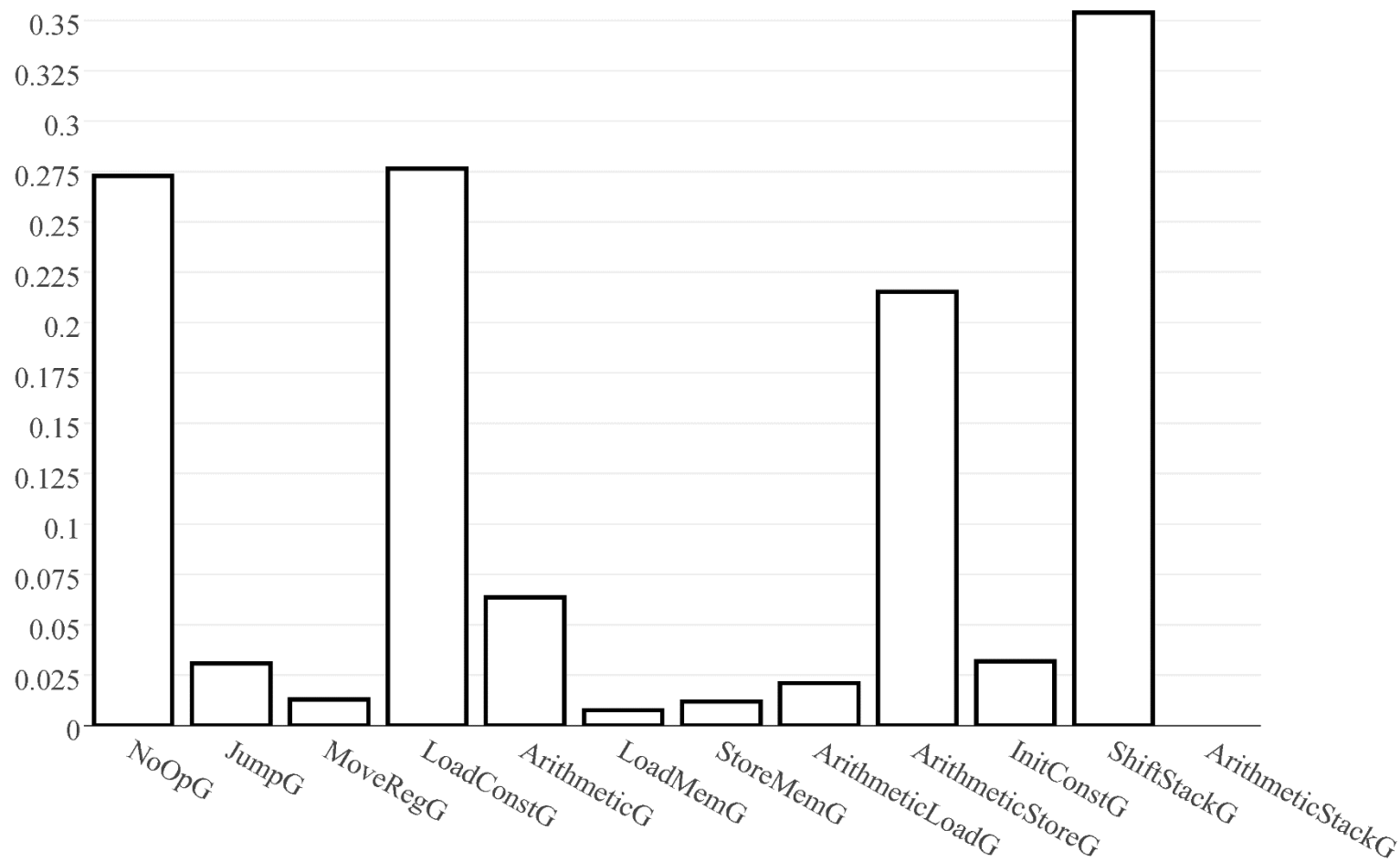
Метод классификации

- На основе полученных значений делается вывод о возможной принадлежности гаджета тому или иному типу
- В результате анализа составляется список всех удовлетворяющих гаджету типов и их параметров (список кандидатов)
- Производится еще несколько запусков процесса интерпретации с различными входными данными, в результате которых из списка кандидатов удаляются ошибочно определенные типы

Статистика по типам гаджетов (x86)



Статистика по типам гаджетов (x86-64)



Результаты

- Реализован алгоритм классификации гаджетов
 - Поддерживаемые архитектуры: x86, x86-64
 - Поддерживаемые форматы исполняемых файлов: elf32, elf64, PE32, PE32+
- На основе результатов классификации гаджетов, найденных в программе zshes была получена работоспособная ROP-цепочка
 - Работает в условии одновременного функционирования защит DEP и ASLR
- Произведена оценка эффективности инструмента ИСП ОБФУСКАТОР
 - Адреса 99.6% классифицированных гаджетов изменились после обфускации

Спасибо за внимание

Слайды будут здесь:
vishnya.xyz/gadgets.pdf