УДК 658.012

В.Н. ГУГНИН, ст. преп. НТУ "ХПИ" (г. Харьков),
Д.В. СОТНИК, вед. программист ЗАО "ИИТ Циклон" (г. Харьков)

АТАКА С ИСПОЛЬЗОВАНИЕМ ПЕРЕПОЛНЕНИЯ БУФЕРА

В статті розглядаються методи підвищення прав користувача та виконання довільного коду з використанням переповнення буферу. Зроблені висновки та наведені рекомендації по ліквідації та упередженню існуючих та потенційних помилок операційної системи що можуть становити загрозу безпеці робочої станції.

In present article were overviewed methods for user rights raising and arbitrary code launching with buffer overflow using. Adducted some conclusion and recommendations about liquidation and forestalling existing and potential operation system errors, which can threat workstation safety.

Постановка проблемы. Переполнение буфера (buffer overflows) – название самой распространенной уязвимости в области безопасности программного обеспечения. Первая атака с применением данной уязвимости использовалась в вирусе-черве Морриса в 1988 году. С тех пор их число увеличивается с каждым годом. В настоящее время можно говорить, что уязвимости, связанные с переполнение буфера, являются доминирующими при удаленных атаках, где обычный пользователь сети получает частичный или полный контроль над атакуемым узлом. Анализ атак и обнаруженных уязвимостей последних лет показывает, что данная проблема является первостепенной.

Анализ литературы. В 1998 году 9 из 13 выпусков CERT (Computer Emergency Response Team site) и по крайней мере половина выпусков 1999 года связаны с переполнением буфера. Отметим, что переполнение буфера присуще также программному обеспечению ряда аппаратных средств. Примером может служить уязвимость принтера HP LaserJet 4500.

Большое количество как идеологических, так и ошибок реализации программного обеспечения делает современные вычислительные системы уязвимыми для повреждения, копирования и изменения хранимых данных [1]. Детальный анализ существующих уязвимостей указывает на то, что внимание безопасности при разработке некоторых сетевых протоколов не уделялось совсем, а соответствующие средства в программные модули встраивались при обнаружении уязвимости [2].

Детальный анализ современных удаленных атак и эпидемий компьютерных вирусов, таких как Lovesun, Mydoom и т.д. (3 самых разрушительных эпидемии за все время в течении последнего года) показал что все массовые и наиболее серьезные и опасные удаленные и локальные атаки строятся на использовании переполнения буфера. Наиболее серьезная уязвимость 2004 года, отнесенная Microsoft к типу Critical, с использованием

которой были связаны 3 наиболее крупных вирусных эпидемии и миллионы взломов рабочих станций была связана с переполнением буфера в RPC DCOM модуле [3]. Однако до сих пор отсутствует классификация описанных атак и способов противодействия по типу, в зависимости от особенностей выполнения внедрения кода.

Цель статьи. В данной статье рассматриваются методы повышения прав пользователя в системе при помощи выполнения произвольного кода с использованием переполнения буфера, как в фрагментах операционной системы, так и в используемом прикладном программном обеспечении. Вводится классификация типов атак с использованием переполнения буфера, рассматриваются способы защиты буфера на этапе разработки приложений.

Основной раздел. Основа атак с использованием этой уязвимости – принцип функционирования операционных систем, где программа получает привилегии и права запустившего ее пользователя или процесса. Таким образом, менее привилегированный пользователь или процесс, который взаимодействует с данной программой, может использовать ее права в своих целях. "Переполнение буфера" делает это возможным. Использование данной уязвимости подразумевает изменение хода выполнения привилегированной программы, например, запуск командной оболочки с правами администратора [4]. Реализации атаки требует решения двух подзадач:

- Подготовка кода, который будет выполнятся в контексте привилегированной программы.
- Изменение последовательности выполнения программы с передачей управления подготовленному коду.

Рассмотрим пути решения подзадачи подготовки кода:

- Подготавливаемый код представляет собой машинные инструкции соответствующего процессора и может передаваться в программу в качестве ее параметра или команды. При этом параметр или команда сохраняется программой в отведенном для этого буфере. Буфер может находится в любой области памяти: в стеке (локальные, автоматические переменные), в динамически распределяемой памяти, в области статических данных. Например, программе, запрашивающей ввод строки, под видом строки может быть передан нужный атакующему код, которому в дальнейшем будет передано управление [5].
- Нужный код не передается в программу, так как он уже присутствует в ней самой или в ее адресном пространстве и требуется лишь его параметризация. Например, подготовка параметра для функции запуска программы. В данном случае атакующему требуется

изменить или сформировать нужный параметр, а не сам код. Параметр также может находится в любой области памяти.

- Если параметризированный код уже присутствует в адресном пространстве программы, то подготовки кода не требуется.

Далее рассмотрим способы передачи управления подготовленному коду. В основе этих способов лежит переполнение буфера, т. е. блока памяти, выделенного под переменную. Переполнение возникает при отсутствии проверки выхода за границы буфера. Таким образом, искажается содержимое других переменных состояния и параметров программы, которые входят в область переполнения буфера. Типы искажаемых объектов-переменных определяет способ передачи управления коду атакующего и могут быть следующими:

Искажение адреса возврата из функции.

Так как вызову функции сопутствует занесение адреса возврата в стек, то при его подмене атакующим, управление передается по заданному им адресу. Здесь используется переполнение буфера локальных переменных функции, которые также создаются в стеке. Частью переменной будет замещен адрес возврата из функции в стеке. Далее, при выполнении инструкции возврата из подпрограммы, управление будет передано по адресу, который образуют соответствующие позиции перезаписанного участка памяти и, в обычной ситуации, будет получено сообщение об ошибке операционной системы. Такие атаки на переполнение буфера получили название "атаки срыва стека" (stack smashing attack).

Искажение указателя функции.

В данном случае атаке подвергаются переменные, содержащие указатели на функции. Эти переменные могут располагаться в любой области памяти, не только в стеке но и в области динамически и статически выделяемых данных. Атакующий организовывает переполнение буфера, которое искажает данные указатели, и при вызове функций по этим указателям управление передается подготовленному коду. Комбинация всех методов подготовки кода и целей переполнения буфера (типа искажаемых структур) определяет виды всех возможных атак по переполнению буфера.

Переполнение буфера происходит, прежде всего, из-за неправильного алгоритма работы программы, который не предусматривает проверок выхода за границы буферов. Также особую роль здесь играет язык программирования Си и его стандартные библиотеки. Так как Си не содержит средств контроля соответствия типов, то в переменную одного типа

можно занести значение другого типа, перезаписывая участок за пределами памяти, отведенной под переменную-назначение.

Таблица

Классификация атак по переполнению буфера [6]

Подготовка кода. Цель переполнения	Внедрение кода	Внедрение параметров
Искажение адреса возврата из функции	Атака "срыв стека"	Атака "срыв стека" с параметризацией
Искажение указателей функций	Атака на указатели функций	Атака на указатели функций с параметризацией
Искажение таблиц переходов	Атака на таблицы переходов	Атака на таблицы переходов с параметризацией
Искажение указателей данных	Атака с искажением указателей данных	Атака с искажением указателей данных с параметризацией

Сложившийся годами стиль программирования, более ориентированный на производительность программ, без выполнения дополнительных проверок, как на этапе проектирования, так и на этапе тестирования, также является причиной распространения данной уязвимости. Рассмотрим способы защиты на этапе проектирования:

Использование неисполнимых буферов.

Суть метода заключается в запрещении исполнения кода в сегментах данных и стека, т.е. параметры сегментов данных и стека содержат только атрибуты записи и чтения, но не исполнения. Например, для реализации неисполняемого стека существуют "заплаты" для ОС Solaris и Linux. Однако ограничение на исполнение данных приводит к проблеме несовместимости. Исполняемый стек необходим для работы многим программам, так как на его основе генерируется код компиляторами, реализуются системные функции операционных систем, реализуется автоматическая генерация кода. Защита с использованием неисполнимых буферов предотвратит только атаки с внедрением кода, но не поможет при других видах атак.

Применение проверок выхода за границы.

В основе данного метода лежит выполнение проверок выхода за границы переменной при каждом обращении к ней. Это предотвращает все возможные атаки по переполнению буфера, так как полностью исключает само переполнение. Проверки выхода за границы переменной опционально реализованы в некоторых компиляторах Си, например, Compaq C, cc в Tru64 Unix, cc в Alpha Linux. Следует отметить, что реализованные проверки ограничены только точными ссылками на элементы массивов, но не производятся для указателей. Однако, у этого решения есть существенный недостаток – значительное (до 30 раз) снижение производительности программы. Другие системы осуществляют проверки при доступе к памяти, выполняя вставки дополнительного объектного кода проверок во все места программы, где есть обращения к памяти. Вставки могут производится как до сборки объектных файлов (Purify) так и после (Pixie). Такие проверки сказываются на производительности с ее уменьшением от 2 до 5 раз и скорее подходят для отладки.

Применение проверок целостности.

Решение, основанное на данном методе, получено благодаря проекту Synthetix [7]. Цель Synthetix – специализация кода для увеличения производительности операционных систем. При этом вводится понятие так называемого квази-постоянства (Quasi-invariant), т.е. состояния среды, которое неизменно в определенных рамках. Такое квази-постоянство позволяет устранить ряд избыточного кода проверки выполнения различных условий. В рамках проекта реализован набор утилит, в том числе обеспечивающих контроль и защиту квази-постоянных состояний среды. К их числу относятся StackGuard и PointGuard. StackGuard предназначен для защиты от всех атак по переполнению буфера с изменением адреса возврата из функции и реализован в виде "заплаты" к gcc. Данная заплата изменяет пролог и эпилог всех функций с целью проверки целостности адреса возврата из функции при помощи так называемого "canary word".

Измененный пролог каждой функции выполняет занесение в стек "canary word", а эпилог проверку содержимого стека, занесенного ранее и, в случае нарушения, останавливает программу с предупреждающим сообщением. При атаке с искажением адреса возврата неизбежно произойдет искажение "canary word", что и будет признаком нарушения целостности. При известном значении "canary word" атакующий может организовать подмену адреса возврата без нарушения целостности. Поэтому "canary word" формируется StackGuard особым образом.

Вторичный стек.

Для предотвращения подмены адреса возврата в прологе каждой функции можно выполнять сохранение этого адреса во вторичном (дополнительном) стеке, а в эпилоге восстанавливать его значение. В случае переполнения буфера и искажения адреса возврата он будет восстановлен эпилогом без выдачи дополнительных сообщений, что впоследствии может привести к аварийному завершению. Атака с подменой указателей функций пресекается путем вставки специального кода перед каждой инструкцией вызова подпрограммы по указателю. Специальный код выполняет проверку того, в каком сегменте расположен адрес, вызываемой подпрограммы. Если это область данных или стека, то программа завершается с ненулевым кодом ошибки. Однако, при такой схеме защиты встает проблема несовместимости с программами, которые содержат исполняемый код в области данных и стека.

Выводы. Рассмотренные методы противодействия атакам по переполнению буфера не выполняют полную автоматическую защиту от всех возможных атак. Ряд атак с искажением указателей данных носят логический характер и не могут быть выявлены в автоматическом режиме. Самая первая атака по переполнению буфера в вирусе-черве Морриса носила именно такой характер. Следует обратить внимание на языки, обеспечивающие проверку и сохранение типов, такие как Java, исключающие переполнение буфера. Однако, не следует забывать, что виртуальная машина Java написана на Си и, таким образом, может иметь уязвимости [8].

Список литературы: 1. *Чекмарева А.Н., Вишнякова Д.Б.* Microsoft (R) Windows 2000: Server и Professional. Русские версии. – СПб.: БХВ – Санкт-Петербург, 2000. – 1056 с. **2.** *Семенов Ю.А.* Протоколы Internet.Энциклопедия. – М.: Горячая линия – Телеком, 2001. – 1100 с. **3.** *Crispin Cowan, Perry Wagle*. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. – Department of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology, 2003. – 235 p. **4.** *Тодд В. Мазерс.* Энциклопедия Windows 2000 для системного администратора. – СПб: Издательский дом "Вильямс", 2001. – 412 с. **5.** *Вильямс А*. Системное программирование в Windows 2000. – СПб.: "ПИТЕР", 2000. – 250 с. **6.** *Steve Bellovin*. Buffer Overflows and Remote Root Exploits. Personal Communications, 1999. – 150 p. **7.** *Семененко В.А., Величкин А.М., Ступин Ю.В.* Операционные системы. – М.: Высшая школа, 1990. – 192 с. **8.** *Саркисян А.А.* Машинонезависимая оптимизация исходных программ. – М.: Радио и связь, 1985. – 208 с.

Поступила в редакцию 29.03.04