Механизмы атак с использованием уязвимостей «Переполнение Буфера».

Гуркин Ю.Н. (технический специалист ООО «ГЛЕГ»)

В настоящее время известно большое количество атак базирующихся на уязвимостях и логических ошибках в программном обеспечении. Так список Common Vulnerabilities and Exposures содержит сейчас более 28000 уязвимостей и пополняется со скоростью 13 новых уязвимостей в день (см. 2). Одними из наиболее опасных и часто встречаемыми удаленными атаками являются атаки базирующиеся на уязвимостях типа «переполнение буфера», о которых и пойдет речь в данном докладе. Помимо описания общих механизмов переполнения буфера, также приведены экспериментальные данные, позволяющие судить о частоте и типах атак наблюдающихся в реальной сети, рассмотрены тенденции в области информационной безопасности.

Инструменты детектирования атак.

Для обнаружения атак с целью их исследования и защиты, используют Системы Обнаружения Вторжений - СОВ. Как и во многих антивирусах, в основе работ многих СОВ лежит прицип детектирования атак с помощью набора правил - сигнатур. Каждая сигнатура как правило соответствует одной атаке. Сигнатура содержит описание конкретного «зловредного» пакета данных. СОВ позволяют надежно детектировать известные производителям СОВ и сетевому сообществу атаки, для таких атак имеются сигнатуры. Кроме того, различные СОВ позволяют с помощью модулей анализа выявлять аномальную активность, а иногда даже обнаруживать атаки с использованием так называемых “0-day” exploits, то есть неизвестные широкому кругу пользователей атаки (см 6). Однако на сегодняшний день не существует ни одной СОВ дающих 100 % гарантию защиты от неизвестных атак.

Экспериментальные данные по наблюдаемым атакам в реальном сегменте сети.

Для выявления атак на функционирующий сегмент сети крупной организации авторами была использована сетевая Система Обнаружения Вторжений с открытыми исходными кодами SNORT (см. 5), с набором правил "VRT certified rules", а также набором правил разрабатываемыми третьими лицами. В СОВ SNORT атакам присваивается приоритет 1, 2, и 3. Атаки с приоритетом 1 - это наиболее опасные атаки на серверные приложения.

Атаки с приоритетами 2, 3 – это чаще всего аномальная активность, которая может свидетельствовать о малоопасной, с точки зрения данной СОВ, атаке, - например, о попытке распространения червя, или разведке доступных сервисов – сканировании.

Защищаемый сегмент сети включал в себя:

более 1000 IP-адресов рабочих станций пользователей, (работающих в основном под ОС Windows), из них в среднем около 200 были активны; несколько серверов, включая сервера http, сервера баз данных SQL, ftp сервер, mail сервер.

В наблюдаемом сегменте сети, SNORT сигнализирует об огромном количестве атак низких приоритетов (2-го и 3-го уровней), тысячах и даже десятках тысяч атак в час. Однако львиная доля этих атак вовсе не являлась атаками. Они лишь свидетельствовали о специфических настройках сетевого оборудования. Среднее же количество детектируемых атак с приоритетом 1, составляло 12 атак в час, включая:

1. Попытки атак на HTTP сервер и установленные web-applications php, jsp, cgi, asp, всего ~8/час
2. Попытки атак баз данных MS-SQL MySQL: ~2/час

3. Попытки атак почтового сервера: ~1/час
4. Попытки атак файлового сервера: ~1/час

Общее количество зарегистрированных разновидностей атак приоритета 1 составило за полгода эксперимента около 50 видов, из них, наиболее часто встречались 12 атак следующих видов:

5-ть разновидностей использования уязвимостей «переполнения буфера»;
1-а разновидность использования уязвимости «дефект форматной строки»;
1-а разновидность с использованием уязвимости “sql-injections”;
5-ть разновидностей использования логических ошибок в программном обеспечении.
Кроме того с помощью инструментов помимо СОВ был зарегистрирован ряд модификаций попыток подбора пароля.

Хотя количество высокоприоритетных атак за час может показаться небольшим, не стоит забывать, что достаточно даже одной успешно проведенной атаки, чтобы нанести существенный вред функционированию сетевой инфраструктуры.

Рабочие станции пользователей также подвергаются угрозам; и даже более того, существует явная тенденция роста интереса атакеров к клиентским приложениям; однако до сих пор именно удаленные атаки на серверные приложения рассматриваются как риск №1. Необходимо отметить, что из 10 разновидностей опасных атак примерно половина – это атаки, базирующиеся на использовании уязвимостей переполнения буфера: «стека» или «кучи».

Разбор механизма атаки переполнения стека - Stack Overflow.

Атаки переполнения буфера основываются на уязвимостях, связанных с отсутствием проверки передаваемых в программу или подпрограмму данных, поэтому может так получиться, что данные, копируемые в буфер, превышают размер самого буфера (отсюда название - buffer overflow). Такие данные запишутся не только в ячейки памяти, выделенные для них, но и в соседние, при этом перезаписывая их содержимое. Такая перезапись может привести к перехвату управления и исполнению внедренного атакером кода (так называемого shell-кода).

Разберем механизм переполнения подробнее на примере следующей простой программы, заносящей в буфер данные введенные пользователем:

```
void fillarray (int a, int b) {
char array[4];
gets(array);
}
main () {
fillarray(1,2);
return 0;
}
```

В программе предполагается, что пользователь введет менее 4 символов. Допустим, пользователь ввел символы «AAAA». При выполнении функции fillarray стек будет заполнен следующим образом:

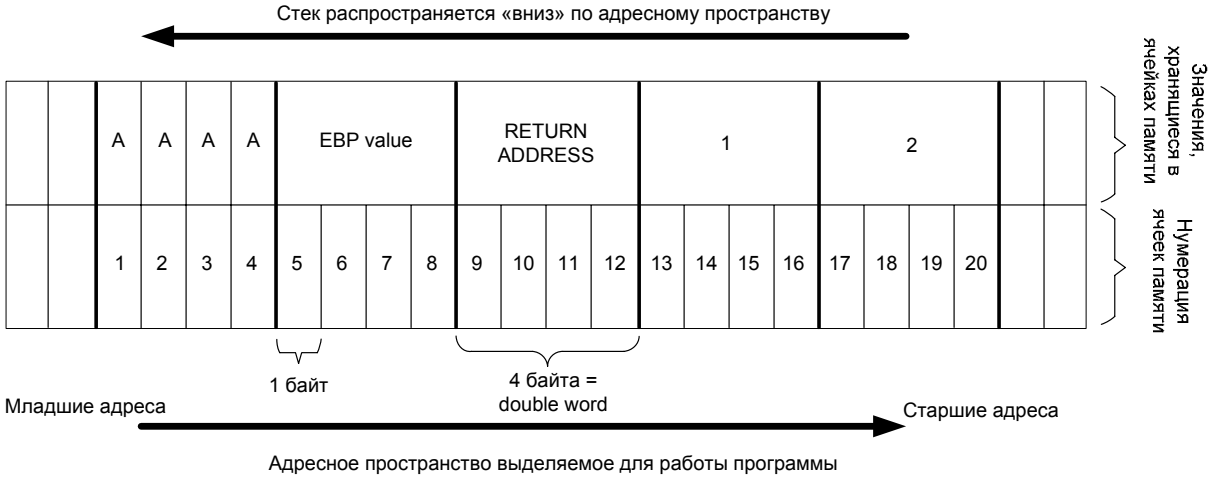

Рис. 2. Заполнение стека

* Адресацию принято записывать в 16-ричном формате, то есть в данном примере адреса ячеек памяти с 1 - 20 могли бы выглядеть, как FFFFFFEC – FFFFFFFC (4 байтовая 16-ричная нумерация адресов в архитектуре IA 32). Значения, хранящиеся в памяти, принято также записывать в 16-ричном формате, так, например, однобайтовому символу A соответствует 16-ричный код 41.

Видно, что в стеке сохраняются в обратном порядке передаваемые в функцию данные, затем адрес возврата из подпрограммы в основную программу, затем значение регистра EBP, затем заполняются пользовательскими данными (а именно символами «AAAA») выделенные для хранения массива array - 4 байта.

Предположим теперь, что нерадивый пользователь ввел не 4 символа A, а 12 символов. Стек будет выглядеть следующим образом:

		A	A	A	A	A	A	A	A	A	A	A	A	1				2					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20		

Рис. 3. Переполнение стека

При переполнении произойдет вот что: Будет заполнено место в памяти выделенное под массив, затем будет перезаписано сохраненное значение EBP, а затем адрес возврата (на его место будет записано «двойное слово», соответствующее 16-ричному представлению символов AAAA, а именно 41414141).

При исполнении программы в данном случае процессор попытается исполнить команду находящуюся по "новому" адресу 41414141 ! атакер при переполнении может записать в адрес возврата такой адрес памяти, в котором содержится внедренный атакером код (“shell-code”),

Разбор механизма атаки переполнение кучи – Heap Overflow:

Как уже говорилось, во время выполнения программы локальные переменные хранятся в стеке, но для глобальных переменных и больших объемов данных требуется другая область памяти для хранения. Такая память выделяется

для записи специальными функциями, имеющими системные привилегии и называется она общим термином «Куча» (Heap).

Разберем механизм атаки типа «Переполнение Кучи», на примере небольшой С-программы, которая последовательно выделяет сегменты памяти размером в 1024 байта для двух переменных buf1 и buf2, с помощью функции malloc, далее копирует заданные пользователем данные в эти переменные, затем вызывает функцию очистки памяти free и завершается:

```
void main (int argc, char **argv)  {
char *buf1,*buf2;
buf1=(char *)malloc(1024);  buf2=(char *)malloc(1024);
strcpy(buf1,argv[1]);  strcpy(buf2,argv[2]);
free(buf2);
}
```

Если пользователь скопирует в буфер buf1 и buf2 меньше 1024 байт, ничего не произойдет. Допустим, пользователь скопировал в обе кучи по 16 символов “A”. Вот как будет выглядеть содержимое памяти выделенной для куч:

Рис 4. Заполнение «Кучи»

Видно, что кучи располагаются подряд друг за другом. В начале каждой кучи первые четыре байта содержат информацию о предыдущем сегменте (если он есть), а именно – его размер в байтах; следующие четыре байта содержат размер текущей кучи.

Теперь предположим, что пользователь скопирует в буфер buf2 больше 1024 байт. Программа и в этом случае завершится без ошибок. Это произойдет потому, что за кучей buf2 память ничем не занята и не содержит никакой служебной информации о сегментах памяти выделенной последующим кучам. Поэтому программа просто «не заметит» переполнения.

А вот как будет выглядеть память, если пользователь скопирует 1028 байт в буфер buf1:

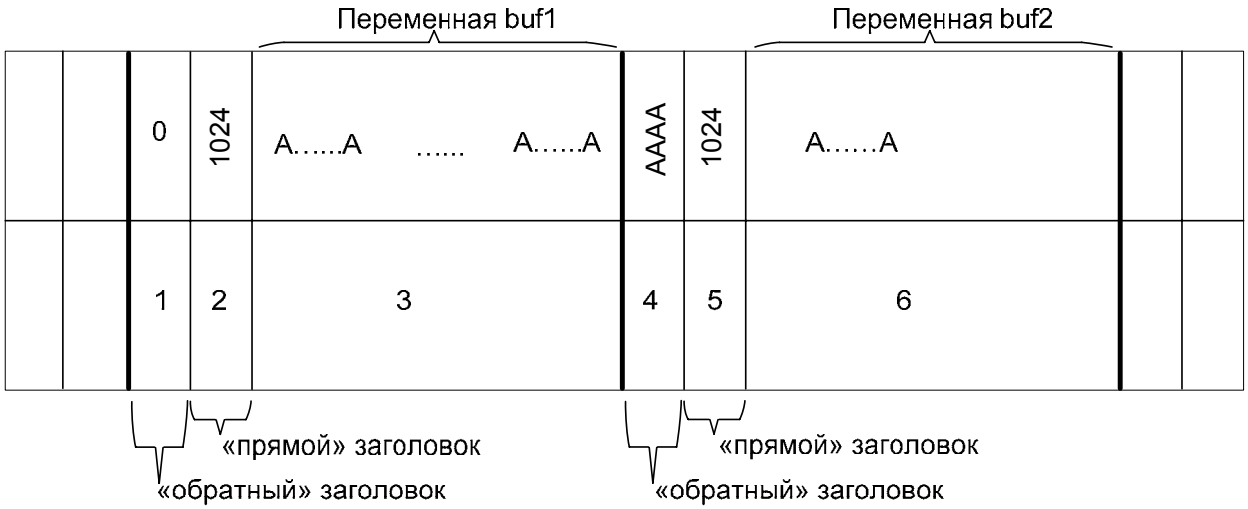

Рис 5. Переполнение «кучи»

Функция free, реализована так, что при ее вызове для очистки памяти buf2 будет предпринята попытка вычислить адрес начала предыдущего сегмента, основываясь на информации из текущего заголовка. Поскольку мы перезаписали этот заголовок, то вычисленный адрес будет неверным и при попытке перехода по нему программа получит сигнал “SIGSEGV ошибка сегментации” и завершится аварийно.

Однако, если будет произведена перезапись так, что программа перейдет по заданному атакером адресу и при этом встретит созданный атакером «фиктивный» сегмент, с «правильным» заголовком, то программа продолжит работать с данными или командами, расположенными в данном сегменте. Так происходит перехват управления при переполнении кучи.

Написание эксплойтов, использующих переполнение кучи гораздо сложнее, чем использование переполнения стека, однако и гораздо многограннее, поэтому защита на уровне ОС от подобных уязвимостей в ближайшие несколько лет врядли будет реализована.

В заключение хотелось бы еще раз отметить рост интереса хакеров к рабочим станциям пользователей и к уязвимостям клиентских программ, так например были выявлены попытки атак таких расширений IE, как ActiveX, атак использующих реализацию протокола PCT 1.0.

Ссылки на использованные ресурсы:

1. http://www.cert.org/advisories/CA-1988-01.html

2. http://nvd.nist.gov/statistics.cfm

3. http://www.securityfocus.com/news/11273

4. Snort 2.1 Intrusion Detection. Andrew Baker, Jay Beale, Brian Caswell, Mike Poore. 2004. Syngress Publishing Inc.

5. The Shellcoder's Handbook: Discovering and Exploiting Security Holes. Jack Koziol, David Litchfield, Dave Aitel, Chris Anley, Sinan "noir" Eren, Neel Mehta, Riley Hassell. 2004. Willey Publishing Inc.

6. http://en.wikipedia.org/wiki/Zero_day