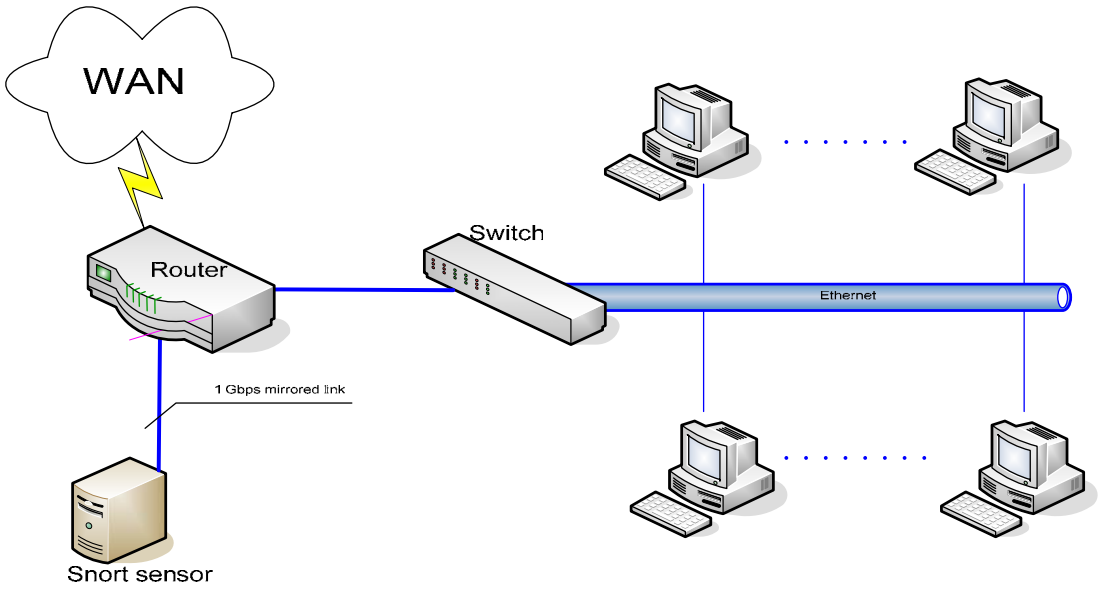
Механизмы атак с использованием уязвимостей «Переполнение Буфера».

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



Защищаемый с помощью Snort сегмент сети

Рис. 1 Схема защищаемого сегмента сети.

Первые удаленные атаки на компьютерные системы были зарегистрированы в конце 80-х годов (см 1.). Реально атаки скорее всего происходили и раньше, со времени создания и расцвета использования языка С.

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). Существуют также атаки базирующиеся на использовании возможностей сетевых протоколов и сетевых ресурсов, как, например, распределенные атаки типа отказ в обслуживании (DDoS), атаки перехвата трафика, подмены серверов и др.

Наиболее опасными считаются удаленные атаки по сети, приводящие к получению удаленного контроля над системой, включая возможность исполнения команд, скачивания и записи файлов. Удаленные атаки используют уязвимости программ, которые работают с данными полученными по сети, это могут быть как серверные программы и ядро ОС, так и служебные программы, например, антивирусы, снифферы, браузеры, мессенджеры. Менее опасны, но весьма чувствительны для пользователей сетевых сервисов удаленные атаки на

серверные программы с целью ограничения доступности или аварийного завершения сервиса (Denial of Service - DoS).

В большом количестве случаев, удаленные атаки с целью получения контроля над системой базируются 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.

Защита от неизвестных атак представляется отдельной весьма актуальной проблемой, так например, нередка ситуация, когда очередная найденная производителем программы уязвимость, была уже известна хакерам (см. 3). Существует не менее нескольких сотен уявимостей, не исправленных производителями популярных серверных и клиентских программ, о которых известно хакерам.

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).

Данная СОВ изначально разрабатывалась для использования квалифицированными специалистами и администраторами крупных сетей, предполагала достаточно сложную настройку для конкретных нужд. Начальные версии SNORT даже поставлялись отдельно от набора правил детектирующих атаки. Возможность и желательность настройки данной СОВ осталась и сейчас, но теперь существует поставляемый вместе с программой, рекомендованный разработчиками SNORT набор правил "VRT certified rules", а также наборы правил разрабатываемые третьими лицами. Упомянутые наборы правил и использовались авторами при анализе актуальных атак.

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 (условная схема рис. 1) 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.

Немалую долю зарегистрированных наиболее опасных атак составляют попытки использования уязвимостей типа "переполнение буфера", в частности стека. Это старый, простой, но до сих пор очень актуальный тип атак.

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:

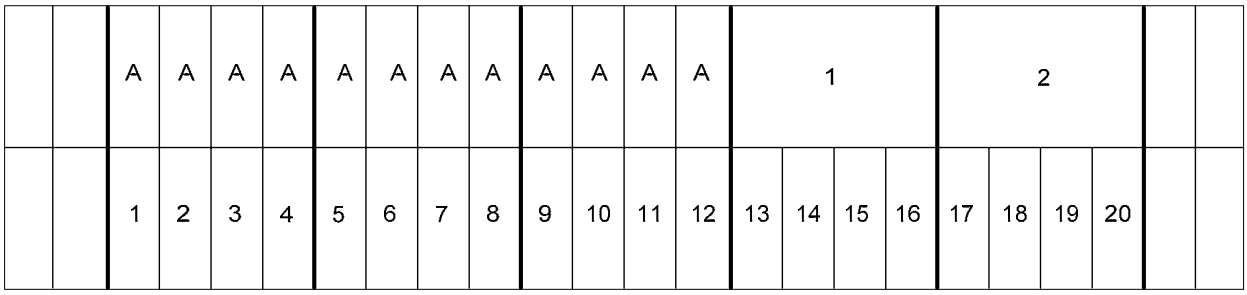


Рис. 3. Переполнение стека

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 ! Поскольку по этому адресу скорее всего ничего не записано, или этот адрес соответствует защищенной области памяти используемой ядром ОС, то ядро ОС сгенерирует прерывание, и произойдет аварийное завершение программы с хорошо знакомой С++ программистам ошибкой - segmentation fault.

Однако атакер при переполнении может записать в адрес возврата такой адрес памяти, в котором содержится внедренный атакером код (“shell-code”), причем после исполнения этого кода и осуществления необходимых атакеру действий, управление может быть передано обратно программе, для обеспечения скрытности взлома.

Таков общий для всех ОС механизм использования наиболее простых, но одновременно актуальных и очень опасных с точки зрения безопасности уязвимостей переполнения Стека. (для изучения дополнительного материала по теме авторы рекомендуют обратиться к книге 4)

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:

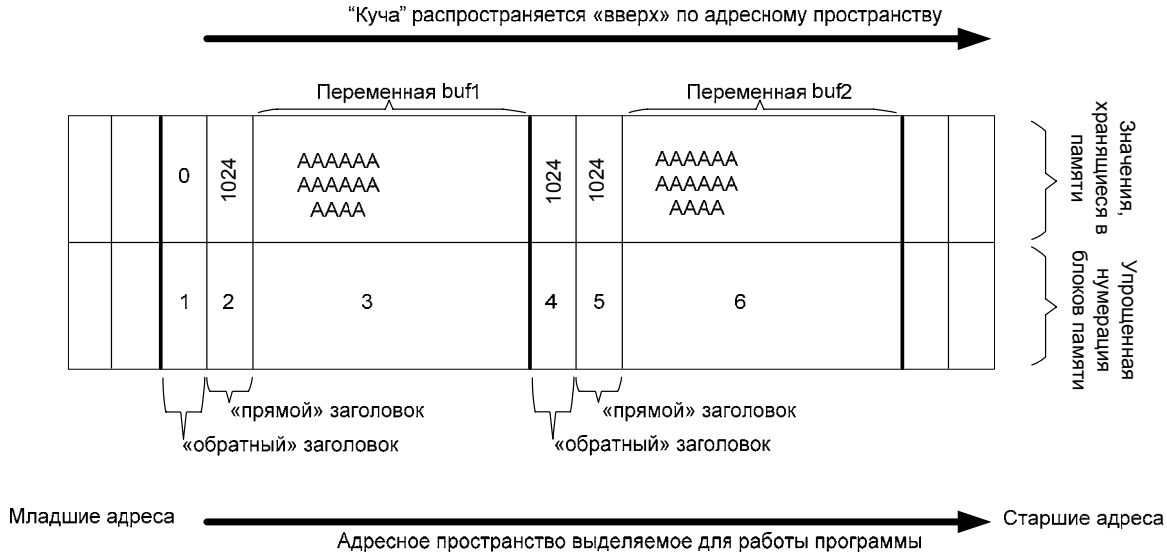


Рис 4. Заполнение «Кучи»

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:

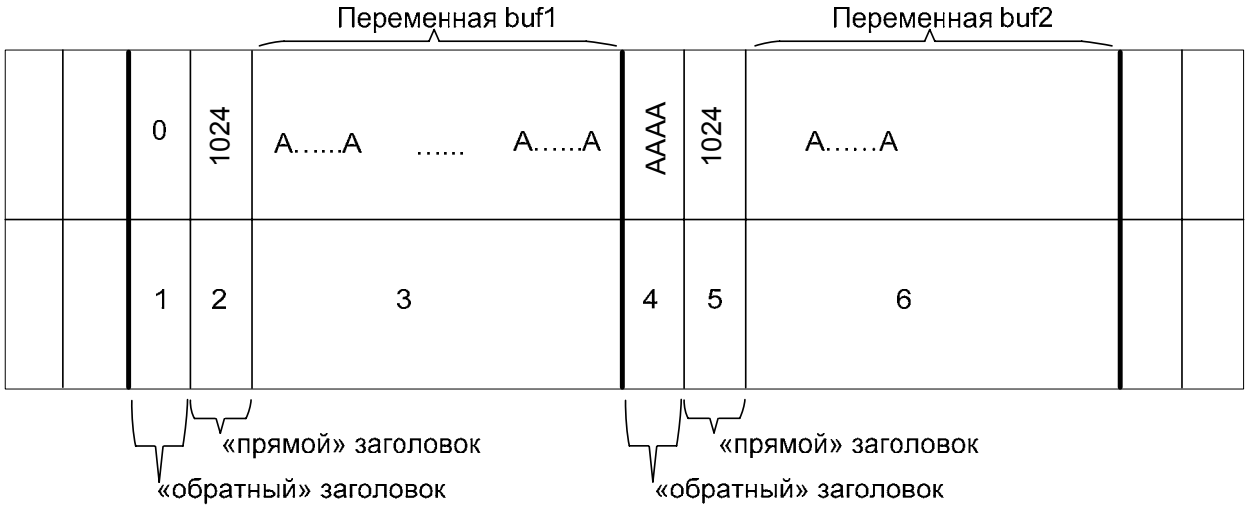


Рис 5. Переполнение «кучи»

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.

Приложение 1: Список детектированных атак с приоритетом 1:

Приведены название атаки (как указано в Снорте), краткое описание атаки и уязвимое ПО (как указано в описании CVE) .

В скобках указан тип уязвимости: (в соответствии с описанием из CVE) stack overflow, heap overflow, sql-injection, logical vulnerability, format string, other.

1 WEB-PHP viewtopic.php access **(sql-injection)**

CVE-2003-0486

SQL injection vulnerability in viewtopic.php for phpBB 2.0.5 and earlier allows remote attackers to steal password hashes via the topic_id parameter.

2 POP3 USER format string attempt **(format string)**

CVE-2003-0391

Format string vulnerability in Magic WinMail Server 2.3, and possibly other 2.x versions, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via format string specifiers in the PASS command.

3 FTP DELE overflow attempt **(heap overflow)**

CVE-2001-0826

Buffer overflows in CesarFTPD 0.98b allows remote attackers to execute arbitrary commands via long arguments to (1) HELP, (2) USER, (3) PASS, (4) PORT, (5) DELE, (6) REST, (7) RMD, or (8) MKD

4 WEB-MISC weblogic/tomcat .jsp view source attempt **(logical)**

CVE-2000-0499

Multiple Vendor URL JSP Request Source Code Disclosure Vulnerability

BEA Systems Weblogic Server 5.1 on all platforms

Apache Software Foundation Tomcat 4.0 on all platforms

5 WEB-IIS asp-dot attempt **(logical)**

CVE-1999-0253

IIS 3.0 with the iis-fix hotfix installed allows remote intruders to read source code for ASP programs by using a %2e instead of a . (dot) in the URL.

6 WEB-MISC ICQ Webfront HTTP DOS **(logical)**

CVE-2000-1078
ICQ Web Front HTTPd (Mirabilis ICQ Web Front is a personal Web server for ICQ users.) allows remote attackers to cause a denial of service by requesting a URL that contains a "?" character.

7 WEB-MISC jigsaw dos attempt **(logical)**
CVE-2002-1052
Jigsaw 2.2.1 on Windows systems allows remote attackers to use MS-DOS device names in HTTP requests to (1) cause a denial of service using the "con" device, or (2) obtain the physical path of the server using two requests to the "aux" device.

8 WEB-MISC PCT Client_Hello overflow attempt **(stack overflow)**
CVE-2003-0719
Win OS almost all versions. PCT protocol stack overflow.

9 SMTP MAIL FROM overflow attempt (**stack overflow**)
CVE-2004-0399
Stack-based buffer overflow in Exim 3.35, and other versions before 4, when the sender_verify option is true, allows remote attackers to cause a denial of service and possibly execute arbitrary code during sender verification.

10 SMTP PCT Client_Hello overflow attempt (**stack overflow**)
CVE-2003-0719
Buffer overflow in the Private Communications Transport (PCT) protocol implementation in the Microsoft SSL library, as used in Microsoft Windows NT 4.0 SP6a, 2000 SP2 through SP4, XP SP1, Server 2003, NetMeeting, Windows 98, and Windows ME, allows remote attackers to execute arbitrary code via PCT 1.0 handshake packets.

11 WEB-MISC encoded cross site scripting attempt (**other**)
Cross-site scripting attack

12 WEB-MISC SSLv2 openssl get shared ciphers overflow attempt (**heap overflow**)
CVE-2006-3738
Buffer overflow in the SSL_get_shared_ciphers function in OpenSSL 0.9.7 before 0.9.7l, 0.9.8 before 0.9.8d, and earlier versions has unspecified impact and remote attack

vectors involving a long list of ciphers.

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