*A. P. Durakovskiy, D. A. Melnikov, V. G. Sergeev*

**Targeted Attacks Using Vulnerability CVE-2013-3897**

*Keywords: vulnerability, malicious code, cyber criminals*

This article provides an overview of the malicious code creation and implementation techniques that were used by cyber criminals to conduct targeted attacks on PCs of residents of the Republic of South Korea in September and October 2013.

*А. П. Дураковский, Д. А. Мельников, В. Г. Сергеев*

# ЦЕЛЕВЫЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ УЯЗВИМОСТИ CVE-2013-3897

## Введение

Актуальность проблемы определяется тем, что одним из наиболее распространенных способов заражения персональных компьютеров (ПК) в настоящее время является использование недостатков (уязвимостей) штатного программного обеспечения (ПО) [1]. Однако наличие уязвимости не означает возможность ее непосредственного и прямого использования. Для этого киберпреступнику, в первую очередь, необходимо создать вредоносный программный код (ВПК), что является весьма сложной задачей, связанной со способностью этого вредоносного кода преодолевать различные средства защиты штатного ПО.

Тем не менее киберпреступниками такой специализированный ВПК был создан [2]. Указанный ВПК, используя свойства и характеристики штатного ПО, может осуществлять предварительную проверку состояния ПК и хранящихся в нем данных, например, с целью идентификации «жертвы», а также скрытно внедрить программную закладку и передавать на нее управление необходимыми для нарушителя процессами и функциями.

В статье рассматриваются современные способы преступного (противоправного) использования уязвимостей W³-обозревателя (*World Wide Web*, W³) «Internet Explorer» на примере ВПК, основанного на уязвимости указанного W³-обозревателя CVE-2013-3897 [3]. Согласно данным, полученным из системы Kaspersky Security Network (KSN) [4], только за две недели октября 2013 г. данный ВПК использовался в попытках заражения ПК более 500 пользователей из Южной Кореи (рис. 1).

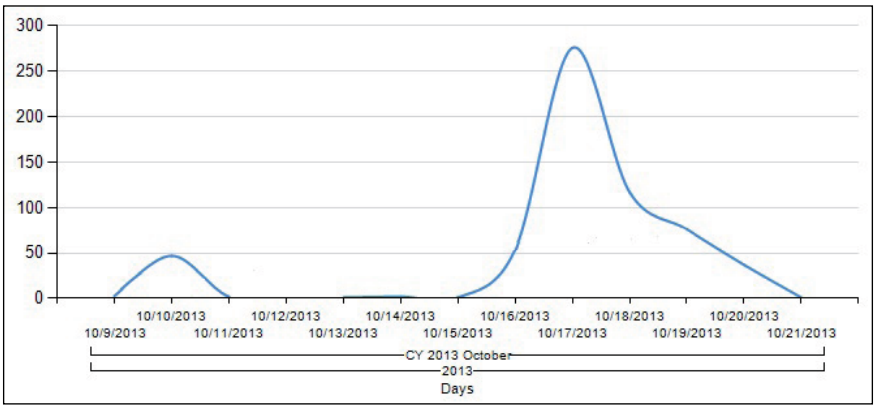


*Рис. 1. График детектирования анализируемого ВПК системой KSN*

Целями атак являлись:

1) кража учетных данных пользователей к четырем популярным интерактивным (*on-line*) играм;

2) получение банковских реквизитов и несанкционированного доступа к банковским счетам этих пользователей.

Данная целевая атака состояла из следующих последовательных этапов: пользователи переходили по ссылке на ложную W³-страницу, содержащую написанный на языке *Javascript* ВПК, который, в свою очередь, использовал уязвимость CVE-2013-3897 в криминальных целях. В результате указанных действий при выполнении процесса W³-обозревателя инициировался необходимый киберпреступнику несанкционированный процесс (на основе ВПК), формирующий скрытое виртуальное соединение. Такое соединение обеспечивало нарушителю несанкционированную фоновую (скрытую) загрузку с его сервера в ПК «жертвы» необходимого для нарушителя ПО, нацеленного на кражу личных данных.

### Анализ атаки на основе уязвимости CVE-2013-3897

Алгоритм атаки с использованием ВПК на основе уязвимости CVE-2013-3897 следующий (рис. 2):

1) пользователь-жертва переходит на сформированную специальным образом W³-страницу, содержащую ВПК. Причем переход может осуществляться как явно, за счет перенаправления с легитимного ресурса, так и с использованием вставки через *iframe* (*i*-кадр) — специализированного контейнера, позволяющего отобразить сторонний документ в контексте уже открытой W³-страницы;

2) во время загрузки вредоносной W³-страницы происходит противоправный захват (заполнение) областей виртуальной памяти (ОВП), отводимых для W³-обозревателя, последовательностью символов, которая представляет собой 4-байтовый адрес ОВП. Этот прием позволяет подготовить своего рода «плацдарм» для размещения части ВПК, написанной в машинном коде (ВПМК), который формирует виртуальное соединение с сервером злоумышленника, по заранее известному адресу виртуальной памяти процесса (ВПП) W³-обозревателя. Данный способ получил название «*heap-spray*» (буквально «*опрыскивать кучу*», «несанкционированный захват (заражение) W³-памяти») [5], то есть заполнение ОВП, выделяемой процессу для динамического размещения данных. Такая ОВП именуется «кучей» (*heap*);

3) ВПК осуществляет предварительную проверку на наличие метки о заражении на ПК «жертвы», а также проверку используемого языка, версии операционной системы (ОС) и W³-обозревателя с целью выполнения ВПМК только на ПК пользователей из Республики Южная Корея и недопущения повторного заражения «жертвы»;

4) в случае прохождения проверки часть ВПК, написанная на *Javascript*, еще раз применяет способ *heap-spray*, но на этот раз с целью расположения ВПМК по известному адресу;

5) ВПК использует уязвимость CVE-2013-3897, в результате чего начинается выполнение ВПМК. Данная уязвимость относится к классу уязвимостей использования ОВП после ее освобождения [6] (*use-after-free*);

6) в начале выполнения ВПМК происходит изменение атрибутов страницы памяти, в которой расположен ВПМК, с целью получения прав на исполнение машинного кода из данной страницы;

7) в конце производится ряд попыток поиска и отключения антивирусного ПО и происходит загрузка основного вредоносного ПО с последующим его запуском.

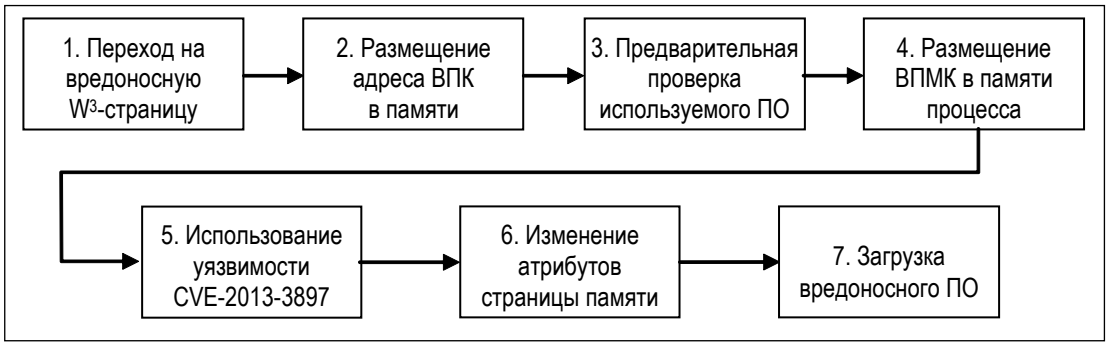


*Рис. 2. Блок-схема алгоритма атаки*

Из схемы атаки видно, что злоумышленниками используются распространенные способы обхода защиты ПО: несанкционированный захват ОВП (*heap-spray*), использование ОВП после ее освобождения (*use-after-free*).

**Способ несанкционированного захвата ОВП**

Способ *heap-spray* используется для доставки ВПК и позволяет поместить данные или программный код по предсказуемому с высокой вероятностью адресу памяти, что достигается за счет особенностей реализации диспетчера памяти в ОС *Windows* [7]:

- адрес начала выделяемой ОВП кратен 8 байтам для 32-разрядных ОС;
- при резервировании больших по размеру ОВП не применяется технология «ассоциативных списков», что позволяет располагать ОВП последовательно;
- виртуальная память процесса остается фрагментированной, несмотря на различные политики управления памятью, используемые диспетчером;
- новые блоки располагаются как можно ближе к началу адресного пространства.

Таким образом, для реализации способа *heap-spray* злоумышленники должны иметь возможность выделять в ВПП множество ОВП большого размера для увеличения вероятности размещения ВПМК по предсказуемому адресу. В случае использования языка *JavaScript* в контексте W3-обозревателя «Internet Explorer» данная задача существенно упрощается, так как при объявлении новых строковых переменных ОВП выделяются для них с помощью WinAPI-функции *HeapAlloc*. Однако строковые переменные в *JavaScript* используют двухбайтовую кодировку, что не позволяет напрямую заполнять ВПП произвольным набором байт. Чтобы обойти данное ограничение, злоумышленники используют *JavaScript*-функцию *unescape*, которая переводит строку из двухбайтовой кодировки в однобайтовую. Строковые переменные в *JavaScript* представляют собой структуру, имеющую тип *Basic string* (BSTR), которая содержит заголовок, строку и байт окончания строки.

Таким образом, учитывая сказанное выше, злоумышленник может точно предсказывать адрес начала ВПМК, размещенного в ВПП путем реализации способа *heap spray*.

**Использование ОВП после ее освобождения**

Подобные уязвимости возникают, как правило, из-за ошибок программистов при инициализации указателей. Классический вариант развития событий при наличии подобной ошибки в ПО выглядит следующим образом:

1) в ВПП выделяется и заполняется определенными данными ОВП;

2) инициализируется указатель, содержащий адрес данной ОВП;

3) выделенная ОВП освобождается и через некоторое время еще раз может быть зарезервирована и заполнена данными, отличными от первоначальных;

4) указатель, инициализированный в п. 2, используется для доступа к ОВП, которая уже была освобождена, вновь занята и изменена.

В большинстве случаев подобное развитие событий приводит к обращению потока к недоступной ячейке памяти, и ПО завершает свою работу с ошибкой.

Однако в некоторых случаях у злоумышленников есть возможность контролировать ОВП, на которую ссылаются подобные указатели. В случае объектно-ориентированного программирования если указатель, инициализированный в объекте, ссылается на таблицу виртуальных методов (ТВМ) [8], то в дальнейшем его значение может быть использовано для вызова одного из виртуальных методов (ВМ) с помощью инструкции вида *CALL [REGISTER]*. Таким образом, если у злоумышленника появляется возможность освободить ОВП, которую занимает ТВМ, то, учитывая арифметические операции с указателем на ВМ, у него появляется возможность выполнить программный код, лежащий по произвольному адресу. В случае использования языка *JavaScript* в W³-обозревателе «Internet Explorer» ВПК, как правило, создает непустой объект, после этого освобождает ОВП, которую занимает ТВМ, с помощью WinAPI-функции *HeapFree* и затем подменяет данную ОВП строкой, содержащей адрес начала ВПМК в ВПП. Далее поток, отвечающий за семантический анализ W³-страницы, производит обращение к ТВМ, в результате чего выполняется ВПМК.

Но из-за того, что указатель, как правило, ссылается на начало ТВМ, ячейка памяти, содержащая указатель на ВМ, может быть заполнена не данными строки, а заголовком BSTR-структуры, значение которой контролировать весьма проблематично. Для решения этой проблемы Питер Ван Икхойте предложил иной способ несанкционированного захвата ОВП, названный *DOM Element Property Spray* [9].

Идея данного способа заключается в заполнении освобожденной области памяти не BSTR-структурами, а другими объектами вроде «кнопки» (*button*), «блока» (*div*) и т. п., в одном из свойств (*src*, *className* и т. д.) которых содержится строка с адресом ВПМК. Подобрав определенный размер строки и количество объектов, злоумышленник способен подменить всю ТВМ нужными ему данными (адресом начала ВПК).

### Анализ уязвимости

Для размещения ВПМК по предсказуемому адресу памяти (0x14141414) ВПК использует способ *heap spray*. Из-за большого количества блоков (в случае данной атаки размер блока — 0x8000 байт) последние блоки будут расположены в области старших адресов памяти последовательно, в силу низкой фрагментации ВПП в области старших адресов. В анализируемом ВПК используется 336 подобных блоков.

```
id_0.onselect=function(e){
Math.atan2(0x999,"before swap");
id_2.swapNode(document.createElement("mark"));
Math.atan2(0x999,"after swap");
}

id_0.onpropertychange=function(e){
Math.atan2(0x999,"before unselect");
var tile=new Array();
for (i=0;i<1000;i++) tile.push(document.createElement("div"));
document.execCommand("Unselect");
for (i=0;i<1000;i++) tile[i].setAttribute("title",str);
Math.atan2(0x999,"after unselect");
}
```

*Рис. 3. Изменение свойств объекта* textarea

Непосредственное использование уязвимости CVE-2013-3897 начинается после полной загрузки W³-страницы и вызова функции *Naver( )*. Данная функция создает два объекта W³-страницы: *textarea* и *address*, причем второй является дочерним объектом первого [10]. Из-за того, что оба объекта являются дочерними по отношению к объекту *body*, при присвоении свойству

*contentEditable* объекта *body* значения *True* создаваемые объекты тоже наследуют это свойство по рекурсии. Далее ВПК вызывает событие *select* по отношению к объекту *textarea*, которое обрабатывается соответствующей функцией. В ходе обработки данного события объект *address* и объект *mark*, созданный в момент выполнения функции, меняются местами [11]. Это является ключевым моментом в работе ВПК (рис. 3).

В результате вызова события *select* поток семантического анализатора $W^3$-обозревателя создает экземпляр класса *DisplayPointer*. После того как ВПК реализует метод *swapNode*, свойство *value* объекта *textarea* автоматически изменяется, и поэтому начинает выполняться функция обработчика события *onpropertychange*, которая создает тысячу пустых объектов *div* и вызывает функцию *Unselect*. Сразу после вызова данной функции еще раз используется способ *heap spray*, который всем только что созданным объектам *div* присваивает свойство *title*, содержащее ASCII-запись «14141414».

В результате работы функции обработчика события *onpropertychange* вызывается ВМ *CDisplayPointer::ScrollIntoView*, который «пытается» изменить позицию экземпляра класса *DisplayPointer*. Однако из-за вызова функции *Unselect* экземпляр класса *CMarkupPointer* ссылается на ОВП, в которой находится не ТВМ, а содержимое свойства *title*, а именно на строку «14141414».

Далее вызывается функция *QIClassID* (см. рис. 4), которая «пытается» выполнить ВМ *CMarkupPointer::QueryInterface*, адрес которого находится в начале ТВМ *CMarkupPointer*, но вместо этого реализуется метод, исполняющий код которого расположен по адресу 0x14141414, то есть там, где находится ВПМК.

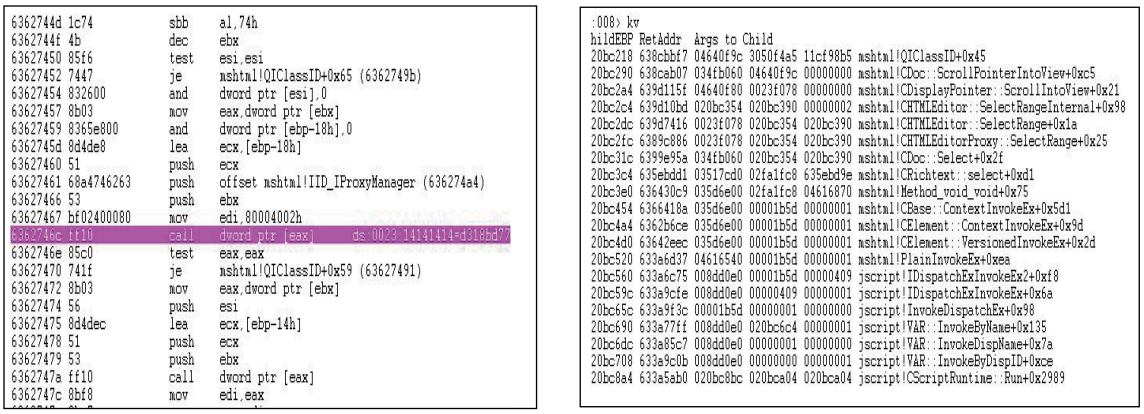


*Рис. 4. Вызов ВПК из подмененной ТВМ*
*(слева — дизассемблированный код, справа — набор вызовов)*

## Заключение

Несмотря на сложность создания вредоносного программного кода, использующего уязвимости штатного программного обеспечения, современные злоумышленники активно разрабатывают новое вредоносное программное обеспечение. Однако способы доставки и внедрения вредоносного программного кода в большинстве случаев остаются неизменными, что позволяет использовать их в качестве сигнатуры в анализаторах вредоносного программного кода и антивирусном программном обеспечении.

## СПИСОК ЛИТЕРАТУРЫ:

1. Kaspersky Lab Global Research And Analysis Team (Great): Kaspersky Security Bulletin 2013. URL: http://media.kaspersky.com/pdf/KSB_2013_EN.pdf (дата обращения: 20.02.2014).
2. *Baumgartner K.* Microsoft Updates October 2013: Older Versions of Internet Explorer Office Silverlight become Ghastly Ghoulish Treehouse of Horrors. URL: http://www.securelist.com/en /blog/ 208214088/ (дата обращения: 20.02.2014).
3. National Vulnerability Database (NVD). CVE-2013-3897. URL: http://W3.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3897 (дата обращения: 20.02.2014).
4. Kaspersky Security Bulletin 2013. Основная статистика за 2013 год. URL: http://xn--80aerbbrjfykgq.xn--p1ai/index.php?p=1&p2=4503 (дата обращения: 20.02.2014).
5. *Van Eeckhoutte P.* Heap Spraying Demystified. URL: https://www.corelan.be/index.php/2011/12/31/exploit-writing-tutorial-part-11-heap-spraying-demystified/ (дата обращения: 20.02.2014).
6. *Bradshaw S.* Internet Explorer Use After Free Aurora Vulnerability. URL: http://www.thegreycorner.com/2010/01/heap-spray-exploit-tutorial-internet.html (дата обращения: 20.02.2014).
7. Русинович М., Соломон Д. Внутренне устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. «Русская Редакция»; СПб.: Питер, 2008 — 990 с.
8. Virtual method table. URL: http:// http://en.wikipedia.org/wiki/Virtual_method_table (дата обращения: 20.02.2014).
9. *Van Eeckhoutte P.* DEPS — Precise Heap Spray on Firefox and IE10. URL: https://W3.corelan.be/index.php/2013/02/19/deps-precise-heap-spray-on-firefox-and-ie10/ (дата обращения: 20.02.2014).
10. Document Object Model. URL: http://en.wikipedia.org/wiki/Document_Object_Model (дата обращения: 20.02.2014).
11. MSDN. swapNode method. URL: http://msdn.microsoft.com/en-us/library/ie/ms536774%28v=vs. 85%29.aspx (дата обращения: 20.02.2014).

## REFERENCES:

1. Kaspersky Lab Global Research And Analysis Team (Great): Kaspersky Security Bulletin 2013. URL: http://media.kaspersky.com/pdf/KSB_2013_EN.pdf (дата обращения: 20.02.2014).
2. *Baumgartner K.* Microsoft Updates October 2013: Older Versions of Internet Explorer Office Silverlight become Ghastly Ghoulish Treehouse of Horrors. URL: http://www.securelist.com/en /blog/ 208214088/ (дата обращения: 20.02.2014).
3. National Vulnerability Database (NVD). CVE-2013-3897. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3897 (дата обращения: 20.02.2014).
4. Kaspersky Security Bulletin 2013. Основная статистика за 2013 год. URL: http://xn--80aerbbrjfykgq.xn--p1ai/index.php?p=1&p2=4503 (дата обращения: 20.02.2014).
5. *Van Eeckhoutte P.* Heap Spraying Demystified. URL: https://www.corelan.be/index.php/2011/12/31/exploit-writing-tutorial-part-11-heap-spraying-demystified/ (дата обращения: 20.02.2014).
6. *Bradshaw S.* Internet Explorer Use After Free Aurora Vulnerability. URL: http://W3.thegreycorner.com/2010/01/heap-spray-exploit-tutorial-internet.html (дата обращения: 20.02.2014).
7. *Russinovich M., Solomon D.* Vnutrennee ustruistvo Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. «Russkaya Redakciya»; SPb.: Piter, 2008 — 990 p.
8. Virtual method table. URL: http:// http://en.wikipedia.org/wiki/Virtual_method_table (дата обращения: 20.02.2014).
9. *Van Eeckhoutte P.* DEPS — Precise Heap Spray on Firefox and IE10. URL: https://W3.corelan.be/index.php/2013/02/19/deps-precise-heap-spray-on-firefox-and-ie10/ (дата обращения: 20.02.2014).
10. Document Object Model. URL: http://en.wikipedia.org/wiki/Document_Object_Model (дата обращения: 20.02.2014).
11. MSDN. swapNode method. URL: http://msdn.microsoft.com/en-us/library/ie/ms536774%28v=vs. 85%29.aspx (дата обращения: 20.02.2014).