Server-Side Request Forgery (SSRF) Prevention in PHP

SSRF is an attack vector that abuses an application to interact with the internal/external network or the machine itself. One of the enablers for this vector is the mishandling of URLs.

Prevention:

a. SSRF attack can be prevented if one of the methods used:
    i. Using static string URL, do not let user input URLs for the server.
    ii. Using white listing if User URL input is required.

Toolkit Scans if project's code has any way of dynamic remote/local calls of other files or websites and notify developer about it.

List of functions which are checked automatically:

1. get_file_content()
2. include()
3. include_once()
4. require()
5. require_once()
6. unlink()
7. highlight_file()
8. show_source()
9. dba_open()
10. bzopen()
11. dba_popen()
12. dbase_open()s
13. fdf_open()
14. rpm_open()
15. dio_open()
16. dbplus_open()
17. fopen()
18. fam_open()
19. dbplus_ropen()
20. imap_open()
21. gzopen()
22. shmop_open()s
23. opendir()
24. openlog()
25. imap_reopen()
26. dbx_compare()
27. zip_open()
28. eio_open()
29. dba_optimize()
30. odbc_binmode()
31. pdf_open_pdi()
32. pdf_open_gif()
33. dbplus_errno()
34. bcmod()
35. db2_close()

36. domentity()
37. putenv()
38. move_uploaded_file()
39. chdir()
40. mkdir()
41. rmdir()
42. chmod()
43. rename()
44. filepro()
45. filepro_rowcount()
46. filepro_retrieve()
47. posix_mkfifo()

Example of wrong code:

```
<?php
   $file_path = $_GET[“user_input”];
   include($file_path);
?>
```

Above code shouldn’t be used, Developer shouldn’t allow user to include or Call any inside or outside files or websites, it should be called statically.

```
The correct code is:
<?php
   include(“config.php”); // static include // not provided by client
?>
```

To learn more, go here:
https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html