Pendeteksian PHP Vulnerabilities Menggunakan Metode *Forward Taint* Data Analisis

ShyffaIlmalliaNoerFhadillah*[1], Parman Sukarno[2], Aulia Arif Wardana[3]
[1,2,3]Telkom University; Jl. Telekomunikasi Jl. Terusan Buah Batu, Sukapura, Kec. Dayeuhkolot, Bandung, Jawa Barat 40257, Telp(022) 7565930
[3]Program Studi Informatika, Telkom University, Bandung
e-mail: *[1]shyffainf@student.telkomuniversity.ac.id,[2]psukarno@telkomuniversity.ac.id,
[3]Aulwardana@telkomuniversity.ac.id

Abstrak

Pemanfaatan website untuk membantu masyarakat baik dalam segi bertukar informasi atau pun melakukan transaksi sudah merupakan hal yang biasa. Namun dalam penggunaan bahasa pemrograman PHP dalam pengembangan website masih sangat rentan terhadap serangan keamanan, contoh serangannya seperti XSS dan SQL Injection. Pada penelitian ini membangun pendeteksian skrip php dengan menggunakan metode Forward Taint data analisis. Sejumlah penelitian telah dilakukan untuk mendeteksi serangan ini. Namun akurasi yang di hasilkan masih sangat rendah. Hal ini disebabkan oleh tingginya nilai false positive yang di hasilkan. Maka dalam menangani hal tersebut, metode forward taint data analisis melakukan pengecekan ganda yang akan mengurangi tingginya nilai false positive yang dihasilkan. Akurasi yang dihasilkan dari peneliian ini mencapai 90%. Hasil ini mengungguli metode lain yang ada.

Kata kunci*—PHP Vulnerability, SQL Injection, XSS, Forward Taint analysis, Regular Expression.*

Abstract

Utilization of websites to help the community both in terms of exchanging information or conducting transactions is common. However, the use of PHP programming language in website development is still very vulnerable to security attacks, for example attacks such as XSS and SQL Injection.This research builds the detection of php scripts using the Forward Taint data analysis method. Research has been carried out to test this attack. However, the accuracy that has been inferred is still very low. This is getting higher because of the false positive generated. So in solving this problem, the forward taint data analysis method performs double checks that will reduce the positive false value generated. Accuracy resulting from this research reached 90%. These results outperform other existing methods.

Keywords*—PHP Vulnerability, SQL Injection, XSS, Forward Taint analysis, Regular Expression*

1. PENDAHULUAN

PHP merupakan salah satu dari bahasa pemrograman web yang sangat popular, bahasa pemrograman PHP digunakan oleh 79,0% dari semua situs web [1]. Banyak aplikasi web telah disediakan untuk layanan *online* dalam beberapa tahun terakhir, seperti perbankan. Tetapi, kerentanan keamanan masih menjadi masalah utama pada aplikasi *web*. Menurut laporan Risk Based Security [2]. Pada tahun 2018 terjadi kerentanan keamanan pada *website* seluruh dunia

mencapai 17.308 serangan, termasuk didalamnya serangan XSS dan *SQL Injection*. Penyebab kerentanan terhadap serangan biasanya terjadi karena adanya Skrip PHP yang tidak difilter, salah satu cara untuk mengatasinya adalah menggunakan *html entities*.

Statis Analisis biasanya digunakan untuk mendeteksi kerentanan keamanan pada suatu program atau *website*, karena Statis Analisis dapat melakukan analisis terhadap suatu Skrip tanpa kita harus menjalankan program atau *website* kita terlebh dahulu [3]. Namun, akurasi pendeteksian PHP *Vulnerability* masih rendah pada beberapa penelitian masih menghasilkan *false positive* yang tinggi. Akurasi yang didapat saat ini hanya mencapai 88% [4].

Maka dalam menangani permasalahan kerentanan keamanan pada suatu Skrip PHP dan rendahnya akurasi pendeteksian pada penelitian ini akan melakukan analisis terhadap Skrip PHP menggunakan metode *Static Forward Taint Data analysis*. Metode *Static Forward Taint Data analysis* dapat melakukan analisis terhadap Skrip PHP tanpa harus menjalankan program tersebut terlebih dahulu atau kita melakukan analisis secara *White Box* serta dapat meningkatkan akurasi pendeteksian kerentanan pada Skrip PHP.

Beberapa langkah untuk melakukan analisis menggunakan *metode Static Forward Taint Data analysis* adalah pertama kita melakukan *tracing* per-baris dan memisahkan antara skrip yang rentan terhadap serangan dan skrip yang tidak rentan terhadap serangan lalu menyimpannya kedalam *array* sesuai kategori atau konteksnya, langkah kedua melakukan *tracing* per-*Array* dan melakukan pengelompokkan jenis serangannya, langkah ketiga melakukan pengecekan kembali pada *array* yang sudah dikelompokkan sebelumnya, apakah ada skrip yang sebenarnya tidak rentan terhadap serangan XSS dan SQL Injection menggunakan Regular Expression. Langkah terakhir adalah melakukan evaluasi dengan menggunakan rumus dan disajikan kedalam tabel.

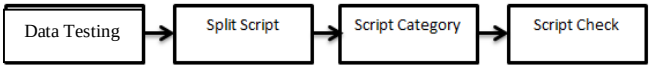
Penelitian ini dilakukan berdasarkan penelitian terdahulu yaitu pada penelitian *A static backward taint data analysis methodfor detecting web application vulnerabilities* [6]. Dimana metode *backward taint data analysis* melakukan pendeteksian dan pembacaan skrip PHP dari baris terakhir menuju baris pertama dan pada penelitian ini dilakukan pendeteksian dan pembacaan skrip PHP dari baris pertama hingga baris terakhir. Hal ini dilakukan agar proses pendeteksian menjadi lebih cepat dan efisien kemudian hasil pendeteksian menjadi lebih akurat, karena sudah ada pola yang disimpan pada *array* atau memori sebelumnya.

.

2. METODE PENELITIAN

2.1 Forward Taint Data Analisis

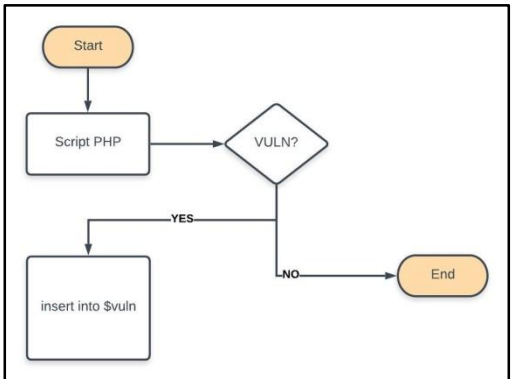
Taint Data Analysis adalah suatu metode yang digunakan untuk memeriksa variabel mana yang dapat dimodifikasi oleh input pengguna. Semua input pengguna dapat berbahaya apabila tidak diperiksa dengan benar. Dan *Forward* disini maksudnya kita melakukan pemeriksaan variabel dari baris pertama, karena kita menggunakan *array* untuk menyimpan variabel yang sudah diperiksa. Untuk proses yang dilakukan akan di jelaskan pada gambar 1.



Gambar 1. Proses Metode *Forward Taint* Data Analisis

2. 1.1Split Script

Split Script disini maksudnya memisahkan Skrip yang rentan terhadap serangan atau tidak rentan terhadap serangan. Proses untuk melakukan Split Script adalah

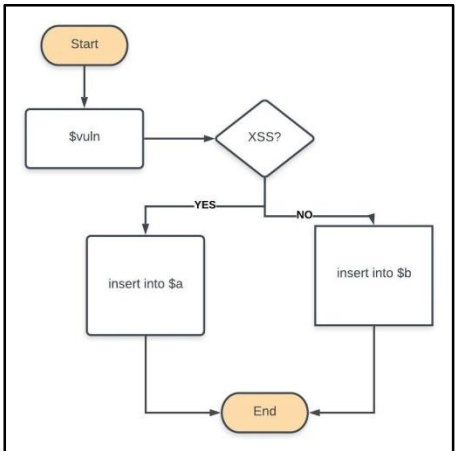


Gambar 2. Proses Split Script

Hasil dari Algoritma ini akan disimpan pada *array* $vuln yang berisikan Skrip PHP dan informasi barisnya.

2. 1.2 Script Category

Script Category ini melakukan pengelompokan Skrip yang rentan terhadap serangan berdasarkan jenis serangannya.



Gambar 3. Proses *Script Category*

Hasil dari proses ini akan disimpan kedalam *array* $a apabila termasuk jenis serangan XSS, dan akan disimpan dalam *array* $b apabila termasuk jenis serangan *SQL Injection*.

2. 1.3 Script Check

Script Check melakukan pengecekan apakah skrip yang sebelumnya sudah kita kelompokan benar benar rentan tehadap serangan. Pengecekan disini menggunakan *Regular Expression* dan melakukan pencocokan dengan *pattern* yang akan kita buat. Langkah – langkah yang dilakukan pada *Skrip Check* adalah membaca *array* per- index lalu dimasukan kedalam *list* atau *array* perkarakter. Langkah kedua adalah masukan kode yang ada pada list atau *array* tersebut kedalam sebuah token. Tanda berakhirnya suatu token adalah ketika bertemu titik koma (;). Langkah terakhir adalah mencocokan token yang sudah kita buat dengan *pattern* serangan yang di buat sebelumnya.

Dari proses ini akan menghasilkan *True positive* (TP) dan *False Negative* (FN). Apabila suatu token mengandung *pattern* yang sebelumnya sudah dibuat, maka akan menambah jumlah dari *True positive* (TP). Namun, apabila suatu token tidak mengandung *pattern* yang sebelumnya sudah dibuat, maka akan menambah jumlah dari *False positive* (FN).

2. 1.4 Pengujian

Penelitian ini menerapkan metode *forward taint data analysis* kedalam aplikasi yang diberi nama FOTA. FOTA adalah aplikasi berbasis bahasa pemrograman Python3 dan dibuat aplikasi berbasis GUI menggunakan PYGTK. Cara kerja aplikasi FOTA ini adalah kita memilih folder mana yang akan kita analisis dan kemudian akan menghasilkan analisis yang berisi informasi tentang jenis serangan dan informasi tentang baris kode yang rentan terhadap serangan.

Proses pengujian menggunakan data test aplikasi php diantaranya Miniblog-1.01, php icalendar 02.4, OA 2012, *Students management system* 2012.

Selanjutnya, melakukan perhitungan menggunakan rumus yang telah dibuat sebelumnya dan akan menghasilkan akurasi yang tinggi [4]. Kemudian parameter yang dianalisis adalah akurasi, rumus yang digunakan:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (1)$$

2. 2 Studi Pustaka

Pada makalah penelitian *Detecting Security Vulnerabilities in object- oriented php programs* [4] ini menyajikan OOPIXY, yaitu sebuah analisis keamanan pada skrip PHP untuk fitur berorientasi objek yang biasanya hanya dibuat untuk fitur terstruktur. Akurasi dengan menggunakan aplikasi ini mendeteksi sampai 88%. Kelebihan lain dari OOPIXY ini adalah dapat mendeteksi skrip php diatas versi 5, dimana PIXI asli hanya dapat mendeteksi sampai PHP versi 4. *Output* dari OOPIXY ini adalah sebuah *file* yang berisi analisis tentang skrip php yang sudah kita *scan* terlebih dahulu dan akan menampilkan akaurasi berapa besar skrip php tersebut terkontaminasi dengan virus – virus yang tersebar.

Pada makalah penelitian PHP *Vulnerability Detection Based on Taint Analysis* [6] ini melakukan pendeteksian kerentanan keamanan pada bahasa pemrograman php menggunakan analisis algoritma *fine-grained taint*. Langkah pertama yang dilakukan adalah mengidentifikasi sumber *codetaint*, ini adalah semua data yang diinputkan pada program eksternal yang dianggap sudah mengandung serangan berbahaya disini menggunakan *Heuristic Strategy*. Terdapat 2 kategori yaitu ketika *user* memasukan *super global code* '$ _GET' dan *sensitive functions* 'mysql_query' maka bisa saja ini termasuk *variable* yang mengandung serangan berbahaya. Setelah itu membuat 2 fungsi secara terpisah yang akan digunakan untuk mendeteksi apakah parameter yang baru dimasukan *user* termasuk kedalam kategori pertama yang ditandai sebagai *user Tainted* atau termasuk kedalam kategori kedua yang ditandai sebagai *secret Tainted*. Setelah mendefinisikan sumber *code taint* maka langkah kedua yang dilakukan adalah melakukan analisis CFG. Dengan cara membangun *maps CFG taint* dan *maps* tersebut berisikan variabel yang rentan akan keamanan dan disimpan dalam bentuk *node*.

Lalu *node-node* tersebut disimpan pada antrian *node*. Lalu inisialisasi *entri node* sebagai *current node*. Untuk mengetahui apakah *node* tersebut rentan akan keamanan atau tidak. Apabila tidak rentan akan keamanan maka letakan *node* tersebut kedalam maps lalu lakukan analisis pada *current node*. Lalu kita mulai untuk melewati *node – node* yang sudah dibuat sebelumnya pada CFG. Setelah melewati semua *node* pada CFG dan telah dilakukan analisis maka akan mengoutputkan suatu hasil yaitu nomor dan jenis kategorinya. Jika tipe *output* adalah *user* Tainted, mungkin ada kerentanan XSS, jika tipe keluarannya adalah *secret Tainted*, di sana mungkin ada kerentanan injeksi SQL.

Pada makalah ini melakukan percobaan pada 16 program yang berbeda dan menggunakan DVWA sebagai *test set*. Makalah ini menghasilkan 4 kelompok dengan tingkat kerentanan yang berbeda-beda yaitu rendah, sedang, tinggi dan tidak mungkin. Namun kekurangan Metode analisis makalah ini saat ini hanya mendukung program berorientasi prosedur, dan penelitian lebih lanjut diperlukan untuk analisis berorientasi objek.

Pada makalah penelitian *A Static Backward Taint Data Analysis Method for Detecting Web Application Vulnerabilities* [5] ini melakukan analisis terhadap skrip php, apakah skrip

tersebut terkontaminasi oleh virus atau tidak. Langkah – langkah yang dilakukan adalah pertama mengumpulkan *sink* dan membangun konteks, yang kedua adalah melakukan *backward tracing* pada variable selama *basic block*, langkah selanjutnya adalah melacak *variable* antara *block*.

Langkah yang terakhir adalah melakukan pelacakan *variable function call*. Maka pada makalah ini dibuat sebuah alat yang bernama POSE dimana alat tersebut mengimplementasikan metode yang akan dibuat seperti diatas dan hasil pengujianpun menyatakan valid bahwa alat POSE ini dapat melakukan pendeteksian pada skrip php atau pada aplikasi web. *Output* pada penelitian makalah ini adalah *abstract syntax tree* (AST), *control flow graph* (CFG) and *call graph* (CG).

Pada makalah penelitian DESERVE: *A Framework for Detecting Program Security Vulnerability Exploitations* [7] ini melakukan proses *embedded monitoring* dalam mengawasi apakah terjadi suatu kerentanan terhadap skrip PHP secara *runtime* dan mengeksekusi *file* exploitasi secara terpisah dengan menggunakan *framework* yang bernama *Deserve*. *Deserve* yang kami gunakan dapat mendeteksi *buffer overflow*, SQL *injection*, dan *cross-site scripting attacks*. *Deserve* tidak memerlukan *dynamic tracking* dan tidak melakukan eksekusi program terlebih dahulu pada *sanbox*. Yang akan kami lakukan pada penelitian selanjutnya adalah meningkatkan *framework deserve* dalam melakukan perlindungan skrip php pada *content sniffing and cross-site request forgery related vulnerabilities.*

Maka disimpulkan penelitian - penelitian yang menggunakan pendekatan yang berbeda-beda tetapi targetnya sama yaitu untuk melakukan pendeteksian kerentanan pada Skrip PHP. Namun hasil akurasinya masih terbilang rendah, penyebab akurasinya masih rendah adalah banyaknya variabel yang tidak terdeteksi sebagai variabel yang rentan terhadap serangan. Maka akan dilakukan peningkatan akurasi dengan menggunakan metode *Forward Static Taint Data Analysis* dan kaitan antara penelitian sebelumnya dengan penelitian pada penelitian ini yaitu pada penelitian ini menggunakan metode *Static Taint Data Analysis* untuk melakukan analisis kerentanan terhadap keamanan suatu Skrip PHP. Pada metode ini akan lebih sedikit variabel yang tidak terdeteksi rentan terhadap serangan karena akan dilakukan pengecekan variabel lebih dari satu kali, maka akurasi yang dihasilkan akan lebih meningkat.

2. 3 Cross-Site Scripting (XSS)

Serangan *Cross-Site Scripting* (XSS) adalah jenis serangan pada inputan *user*, di mana penyerang menginputkan suatu skrip ke sebuah aplikasi php. Serangan XSS dapat terjadi karena aplikasi tersebut tidak melakukan validasi skrip terlebih dahulu.

Penyerang dapat menggunakan XSS untuk mengirim skrip berbahaya kepengguna yang lain. Karena pengguna menganggap skrip tersebut berasal dari sumber terpercaya maka pengguna tetap mengakses skrip tersebut.

Maka pada penelitian ini dilakukan pengamanan skrip dari serangan XSS, dan akan dijelaskan pada Tabel 1.

Tabel 1. XSS Vulnerability

No.	*Vulnerability*	*Function Name*	*Protection Function*
1.	XSS	Echo	Htmlentities, htmlspecialchars
2.	XSS	Print	Htmlentities, htmlspecialchars
3.	XSS	Printf	Htmlentities, htmlspecialchars

2. 4 SQL Injection

Serangan SQL *Injection* adalah jenis serangan keamanan pada sisi *client* dengan cara memodifikasi perintah SQL. Pengaruh dari serangan ini sangat berbahaya karena akan menyerang data sensitif yang disimpan di dalam *database*. Maka pada penelitian ini dilakukan pengamanan skrip dari serangan SQL *Injection*, dan akan dijelaskan pada Tabel 2.

Tabel 2. SQL *Injection Vulnerability*

No.	*Vulnerability*	*Function Name*	*Protection Function*
1.	SQL Injection	mysql_query	mysql_real_escape_string
2.	SQL Injection	mysqli_multi_query	mysql_real_escape_string
3.	SQL Injection	mysqli_send_query	mysql_real_escape_string
4.	SQL Injection	mysqli_master_query	mysql_real_escape_string
5.	SQL Injection	mysql_unbuffered_query	mysql_real_escape_string
6.	SQL Injection	mysql_db_query	mysql_real_escape_string
7.	SQL Injection	mysqli::real_query	mysql_real_escape_string
8.	SQL Injection	mysqli_real_query	mysql_real_escape_string
9.	SQL Injection	mysqli::query	mysql_real_escape_string
10.	SQL Injection	mysqli_query	mysql_real_escape_string
11.	SQL Injection	*array*Query	Prepare
12.	SQL Injection	Query	Prepare
13.	SQL Injection	queryExec	Prepare
14.	SQL Injection	singleQuery	Prepare
15.	SQL Injection	querySingle	Prepare
16.	SQL Injection	Exec	Prepare
17.	SQL Injection	Execute	Prepare
18.	SQL Injection	unbufferedQuery	Prepare
19.	SQL Injection	real_query	Prepare
20.	SQL Injection	multi_query	Prepare
21.	SQL Injection	send_query	Prepare
22.	SQL	cubrid_un	cubrid_real_escape_st

	Injection	buffered_ query	ring
23.	SQL Injection	cubrid_qu ery	cubrid_real_escape_st ring
24.	SQL Injection	mssql_que ry	mssql_escape

2. 5 Regular Expression

Regular Expression adalah sebuah kumpulan *string* yang disusun dan digunakan untuk mencocokan sebuah *text* menggunakan pola tertentu. Pada penelitian ini dibuat sebuah *pattern* yang berfungsi untuk pengecekan skrip pada proses terakhir.

Pada penelitian ini kami membuat pola menggunakan *regular expression* dan digunakan untuk mengecek kerentanan terhadap skrip php.

3. HASIL DAN PEMBAHASAN

Gambar 4. Hasil Penelitian

Aplikasi ini berjalan dengan sukses pada semua versi Windows dan Linux. Dari hasil pengujian tersebut menghasilkan 20 *code* PHP yang rentan terhadap XSS dan SQL *Injection.* Dan setelah dilakukan pengecekan dan dibandingkan dengan hasil yang sudah kita dapat maka menghasilkan *True positive* (TP) sebanyak 20, *False positive* (FP) sebanyak 7, *True Negative* (TN) dan *False Negative* (FN) sebanyak 0.

False positive disini berasal dari -> execute ($result) *code* ini terdeteksi sebagai serangan SQL *injection* dengan jenis PDO SQL *Injection* karena mengandung *execute*, padahal maksud dari *code* tersebut adalah melakukan rekursif dengan memanggil *function execute* yang sudah dibuat sebelumnya.

```
Folder 54: C:/Users/sifa user/Downloads/dataset/DVWA-master\external\phpids\0.6\lib\IDS\vendors\htmlpurifier\HTMLPurifier\Strategy
================================================================================================================================
Potential vulnerability found : SQL Injection
Line 16 in C:/Users/sifa user/Downloads/dataset/DVWA-master\external\phpids\0.6\lib\IDS\vendors\htmlpurifier\HTMLPurifier\Strategy/Composite.php
Code : ->execute($tokens, $config, $context)
Declared at Line 17 : $tokens = $strategy->execute($tokens, $config, $context));
```

Gambar 5. Kode *False Positive*

Pada penelitian ini akan dilakukan perbandingan dengan menggunkan aplikasi POSE. POSE adalah sebuah aplikasi dari implementasi metode *static backward taint data analysis*. Perbandingan akan ditampilkan pada tabel 3.

Tabel 3. Hasil Perbandingan POSE dan FOTA

Programs	Vulnerabilities confirmed	Vulnerabilities Reported (POSE)	Vulnerabilities Reported (FOTA)	False Positive
Miniblog-1.0.1	1	2	2	1
phpicalendar-2.4	2	6	4	2
OA2012	4	10	6	2
Students management system 2012	6	10	8	2
Total	13	28	20	7

Kemudian setelah melakukan perbandingan, langkah selanjutnya adalah melakukan perhitungan menggunakan rumus yang dibuat sebelumnya, Akurasi yang dihasilkan adalah 90%.

Berdasarkan tabel 3, kita dapat melihat bahwa *false positive* yang dihasilkan oleh aplikasi FOTA lebih sedikit dari yang dihasilkan oleh aplikasi POSE. Ini disebabkan oleh pengecekan ulang yang dilakukan oleh aplikasi FOTA.

4. KESIMPULAN

Kesimpulan dari penelitian ini yaitu analisis skrip php menggunakan *forward taint data analysis* adalah sesuai dengan tujuan pembuatan pada penelitian ini yaitu untuk meningkatkan akurasi metode ini dapat meningkatkan akurasi yang sebelumnya hanya mencapai 88% sedangkan dengan menggunakan metode ini dapat mencapai akurasi sampai 90%, akurasi ini dapat meningkat karena dalam metode ini kita melakukan pengecekan ganda, yang pertama memeriksa apakah dia rentan atau tidak, yang kedua memeriksa apakah dia benar-benar rentan atau tidak menggunakan pola yang dibuat menggunakan *Regular Expression*. Meningkatnya akurasi ini disebabkan oleh sedikitnya *False positive* yang di hasilkan.

5. SARAN

Adapun saran untuk penegembangan aplikasi yaitu kami akan memperbanyak jenis serangan yang dapat diserang.

UCAPAN TERIMA KASIH

Pertama dan yang utama, saya ingin berterima kasih kepada Allah SWT atas rahmat-Nya yang telah memberikan kelancaran kepada saya untuk menyelesaikan penelitian ini. Juga untuk orang tua saya atas dukungan luar biasa mereka dalam proses ini.

Saya ingin mengucapkan terima kasih juga kepada Laboratorium Foresty telah membantu proses penelitian saya. Terakhir saya ingin mengucapkan terima kasih kepada semua teman saya yang selalu mendukung saya dalam semua situasi.

DAFTAR PUSTAKA

[1] W3techs, "*Usage Statistics and Market Share of PHP for Websites*," 2019. [Online]. Available: https://w3techs.com/technologies/details/pl-php/al. [Accessed: 29-Mar-2019].

[2] R. B. Security, "*lnerability Quick View 2016 Year End*," 2017. [Online]. Available: https://pages.riskbasedsecurity.com/2016-ye- nulnquickview. [Accessed: 29-Mar-2019].

[3] C. Dimastrogiovanni and N. Laranjeiro, "*Towards Understanding The Value of False Positives in Static Code Analysis*," *Proc. - 7th Latin-American Symp. Dependable Comput. LADC 2016*, vol. 685, pp. 119–122, 2016, doi: 10.1109/LADC.2016.25.

[4] M. Nashaat, K. Ali, and J. Miller, "*Detecting Security Vulnerabilities in Object-Oriented PHP Programs*," *Proc. - 2017 IEEE 17th Int. Work. Conf. Source Code Anal. Manip. SCAM 2017*, vol. 2017-October, pp. 159–164, 2017, doi: 10.1109/SCAM.2017.20.

[5] X. Yan, H. Ma, and Q. Wang, "*A Static Backward Taint Data Analysis Method for Detecting Web Application Vulnerabilities*," *2017 9th IEEE Int. Conf. Commun. Softw. Networks, ICCSN 2017*, vol. 2017-January, pp. 1138–1141, 2017, doi: 10.1109/ICCSN.2017.8230288.

[6] K. Cao, J. He, W. Fan, W. Huang, L. Chen, and Y. Pan, "*PHP Vulnerability Detection Based on Taint Analysis*," *2017 6th Int. Conf. Reliab. Infocom Technol. Optim. Trends Futur. Dir. ICRITO 2017*, Vol. 2018-January, pp. 436–439, 2018, doi: 10.1109/ICRITO.2017.8342466.

[7] A. Mohosinaand and M. Zulkernine, "*DESERVE: A Framework for DEtecting Program Security Vulnerability Exploitations," Proc. 2012 IEEE 6th Int. Conf. Softw. Secur. Reliab. SERE 2012*, pp. 98–107, 2012, doi: 10.1109/SERE.2012.22.