Jurnal RESTIA Volume 1, Nomor 1, Agustus 2023 **Hal 1-11**
e-ISSN : 2988-5663

OVERCOMING PROBLEMS COMBINATION OF SERVER AND CLIENT SIDE ENCRYPTION ON WEB CODE SECURITY USING THE SLICE CODE METHOD

Kamarudin [1], Fajar Akbar Maulana [2]
[12]Prograsm Studi Informatika, Fakultas Teknik, Universitas Muhammadiyah Banjarmasin
Jl. S. Parman Kompleks RS Islam, Ps. Lama, Kec. Banjarmasin Tengah, Kota Banjarmasin, Kalimantan Selatan 70114
Kamarudin.skom@gmail.com , namasayafajarakbar@gmail.com

ABSTRACT
Web security protection that is carried out by some web developers is to protect the source code of web applications by encrypting the source code, both the source code that runs on the web server or the source code that runs on the client browser. Combining these two encryption technologies is a problem for web developers. If using server side encryption, the client side source code is not protected. However, if you use client-side encryption, the server source code becomes unreadable by the web server. This study aims to overcome the problem of the combination of these two encryption technologies, so the Slice Code method is used, which is a method that the researchers themselves have developed and used to encrypt server and client source code simultaneously and combine the two. With this method, the original source code is separated into 2 parts, then the pure client source code is encrypted using client encryption technology, then embedded into the pure server source code. The next step is to encrypt the pure server source code. The result is that web applications can run perfectly by having source code encryption from both sides, namely the server side and the client side.

Keywords : encryption, source code, slice code, web security, web encryption

I. INTRODUCTION

Perkembangan pemrograman web sudah sedemikian pesatnya dengan beragam teknologi web yang ditawarkan, mulai dari teknologi framework yang membuat tampilan aplikasi web menjadi lebih interaktif dan responsive, hingga framework yang memberikan kemudahan dalam pengembangan dan pengelolaan data. Oleh sebab itu, tidaklah mengherankan jika aplikasi web modern saat ini selain akan memberikan tampilan yang menawarkan pengalaman baru untuk pengguna dalam pemanfaatan web, kecepatan proses penyimpanan dan loading data, hingga kecepatan dalam pembuatan dan pengembangan aplikasi web itu sendiri [1].

Meskipun demikian, keamanan web merupakan hal yang harus menjadi fokus dalam pengembangan aplikasi web. Keamanan web menjadi faktor utama untuk membuat aplikasi web yang membutuhkan pemrosesan data dan penyimpanan data, dan transter data antar komputer. Banyak serangan online yang dilakukan oleh orang-orang yang tidak bertanggung jawab dengan latar belakang dan tujuan yang berbeda. Mulai dari pencurian data [2][3][4][5], pengrusakan data [6] [7][8], hingga serangan ringan yang menyebabkan aplikasi web menjadi tidak berfungsi sebagaimana mestinya [4][9]. Serangan ini memiliki beberapa jenis yang popular, antar lain Broken Access Control yaitu serangan yang berfokus pada kecacatan terhadap kebijakan yang diberlakukan kepada pengguna terhadap akses apa saja yang diizinkan [10], Cryptographic Failures yaitu serangan ini mentargetkan pada kegagalan atau kekurangan yang berkaitan dengan kriptografi [11], Injection yaitu serangan yang memanfaatkan kelemahan pada data yang diberikan pengguna yang tidak dilakukan validasi atau dibersihkan oleh aplikasi [12][13], Insecure Design yaitu serangan yang ditujukan untuk menghilangkan profil risiko bisnis yang melekat pada aplikasi atau sistem yang sedang dikembangkan [14], Security Misconfiguration yaitu serangan yang berfokus pada kesalahan konfigurasi sistem atau aplikasi yang biasanya diabaikan begitu saja oleh pengembang web [15][16], Vulnerable and Outdated Components yaitu serangan yang mentargetkan pemanfaatan komponen yang tidak dikelola dengan baik dari pihak ketiga oleh pengembang web [17], Identification and Authentication Failures yaitu serangan yang mentargetkan pada kegagalan identifikasi, validasi sertifikat yang tidak memiliki kecocokan dengan host serta otentikasi yang tidak benar [18] [19], Software and Data Integrity Failures yaitu serangan yang mentargetkan aplikasi web yang tidak melindungi dari pelanggaran integritas terhadap plug-in yang digunakan [20], Security Logging and Monitoring Failures yaitu serangan yang mentargetkan pada pencatatan dan pemantauan sistam yang digunakan pada pengembangan aplikasi web yang seringkali pelanggaran tidak dapat dideteksi [21], dan Server-Side Request Forgery, yaitu serangan ini mentargetkan cacat pada SSRF yang terjadi dimana aplikasi web acap kali mengambil sumber daya dari tempat lain tidak mevalidasi URL yang digunakan [22]. Ini memungkinkan penyerangan ke sistem bahkan ketika dilindungi oleh firewall, vpn atau lainya sesuai dengan access control list (ACL), penyerang memanfaatkan pemaksaan terhadap aplikasi web untuk mengirim permintaan yang dibuat ke tujuanyang tidak terduga.

Perlindungan keamanan web yang sebagai developer web lakukan adalah dengan melindungi source code aplikasi web dengan cara melakukan enkripsi terhadap source code, baik kode sumber yang berjalan melalui server web atau kode sumber yang berjalan di browser client. Source code terenkripsi di server adalah source code yang berada di server yang kode aslinya tidak dapat dilihat dan dirubah oleh siapa pun, karena sudah diacak dengan karakter-karakter yang tidak lazim dan terlihat asing, sehingga tidak diketahui instruksi-instruksi apa yang tertulis di source aslinya. Saat aplikasi web terikripsi ini dijalankan di browser web, aplikasi ini dapat berjalan sebagaimana mestinya di browser, tidak ada perubahan apa pun. Hal ini disebabkan web server secara otomatis merubah instruksi source code terenkripsi di server menjadi format dokumen html yang kemudian diteruskan ke browser client. Inilah yang menjadi kelemahan dari source code terenkripsi di server, yaitu tidak mampu melakukan enkripsi pada sisi client, source code client -html- masi bisa dilihat melalui browser, dan ini berbahaya karena bisa menjadi pintu masuk bagi serangan ke server [23][24].

Meskipun demikian, bisa saja dilakukan enkripsi terhadap source code client -html- dengan menggunakan berbagai algorima dan tool yang tersedia, hanya saja hal ini justru membuat source code server menjadi tidak berfungsi saat dibaca oleh web server dikarenakan perbedaan dalam pembacaan source code. Hal umum yang biasa dilakukan oleh developer web adalah menggabungkan antara source code server dengan source code client -html- menjadi satu kesatuan file, baik di file inputan atau file untuk menampilkan data di browser [25].

Menggabungkan kedua teknologi enkripsi ini menjadi masalah tersediri bagi kalangan developer web. Jika menggunakan enkripsi server side, maka source code client side tidak terlindungi. Namun, jika menggunakan enkripsi client side, maka source code server menjadi tidak terbaca oleh web server. Untuk mengatasi masalah kombinasi kedua teknologi enkripsi ini, maka digunakanlah metode Slice Code (kode irisan), yaitu metode yang peneliti sendiri kembangkan dan gunakan dalam mengenkripsi source code server dan source code client secara bersamaan serta mengkombinasikan keduanya. Dengan metode ini, source code asli dipisahkan menjadi 2 bagian, source code server murni dan source code client murni, kemudian source code client murni dienkripsi menggunakan teknologi enkripsi client, lalu diembed ke source code server murni. Langkah selanjutnya adalah dengan mengekripsi source code server murni. Hasilnya aplikasi web dapat berjalan sempurna dengan memiliki enkripsi source code dari kedua sisi, yaitu sisi server dan sisi client.

Berdasarkan penuturan Mohammad Harun Alfirdaus dan kawan-kawan dalam artikelnya yang diterbitkan tahun 2023, menuturkan permasalahan perkembangan teknologi informasi yang berdampak pada kemudahan akses terhadap media komunikasi telah mempengaruhi keamanan informasi dan pesan yang disampaikan melalui media tersebut.. Untuk mengatasi permasalahan keamanan tersebut, penulis menggunakan metode enkripsi Caesar yang diterapkan di bahasa pemrograman PHP. Awalnya, penulis memulai dengan merancang sebuah aplikasi web yang menggunakan bahasa pemrograman PHP dan markup language HTML. Dari aplikasi tersebut, kemudian dibuat sebuah fitur yang memberikan perlindungan pada pesan melalui penerapan algoritma pengacak Caesar. Salah satu cara yang digunakan adalah dengan mengubah karakter-karakter dalam pesan asli menjadi ciphertext. Dalam prosedur ini, teks biasa mengalami perubahan melalui penggunaan kunci yang merupakan hasil dari perhitungan urutan abjad pada teks tersebut. Hasilnya adalah hanya pihak berwenang saja yang dapat berinteraksi dengan pesan pribadi secara bebas [26].

Sementara itu, Ahmad Shofi dan kawan-kawan menuturkan dalam artikelnya yang berjudul "Enkripsi dan Deskripsi dengan Metode Encryption System (DES) dengan Menggunakan Bahasa Pemrograman PHP" menjelaskan bahwa dengan menggunakan metode enkripsi DES maka masalah kebutuhan keamanan data sangat diperlukan agar terhindar dari serangan pencurian data. Hasilnya adalah data menjadi lebih aman dengan antisipasi celah-celah keamanan kata sandi dari para hacker [27].

Meskipun demikian, Anggi Elanda dan Robby Lintang Buana dalam artikelnya mengatakan apakah metode Open Web Application Security Project (OWASP) dapat digunakan untuk mendeteksi keamanan pada sistem informasi berbasis web. Metode yang digunakan adalah dengan membuat tinjauan sistematis melalui 3 literatur dari beberapa sumber penerbit, lalu membuat perbandingan hasil OWASP dan tingkat keamanan server web dari sumber penerbit [28].

II. RESEARCH METHODS

Sebagaimana dipapakan sebelumnya, bahwa metode yang digunakan untuk mengatasi masalah kombinasi enkripsi source code pada aplikasi web, baik dari sisi server atau dari sisi client adalah dengan menggunakan metode Slice Code dengan tahapan sebagai berikut:

1. File Aplikasi Web Murni. Pada bagian ini terdapat ilustrasi file aplikasi web yang berisi serangkaian kode program web yang tercampur antara kode program server dan kode html.

Fig. 1. Isi File0

2. Pembagiaan -slice- kode sumber program menjadi file terpisah. Proses ini memisahkan antara file kode sumber server dan kode sumber html.

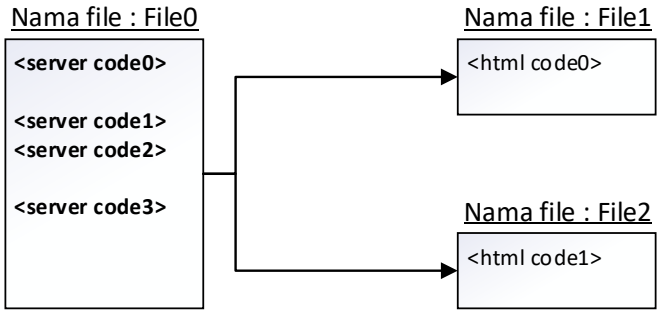

Fig. 2. Isi File0 Terpisah

3. Meng-enkripsi masing-masing kode sumber html : File1 dan File2 menggunakan metode enkripsi client side.

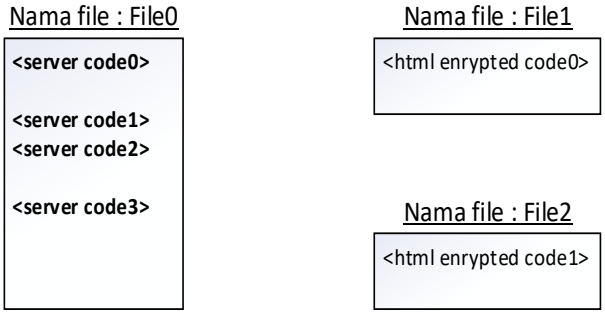

Fig. 3. Isi File1 dan File2 Terenkripsi

4. Meng-include-kan File1 dan File2 yang sudah di-enkripsi ke File0 yang belum di-enkripsi. Proses ini menggunakan fungsi include file yang disediakan oleh kode server. Sehingga jika dilakukan proses enkripsi pada File0 tidak akan mempengaruhi File1 dan File2 yang sudah di-enkripsi.

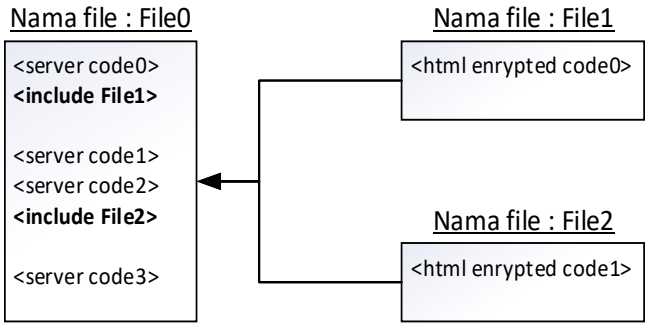

Fig. 4. Isi File1 dan File2 Disisipkan

5. Melakukan enkripsi pada File0 sebagai langkah akhir proses enkripsi kode sumber

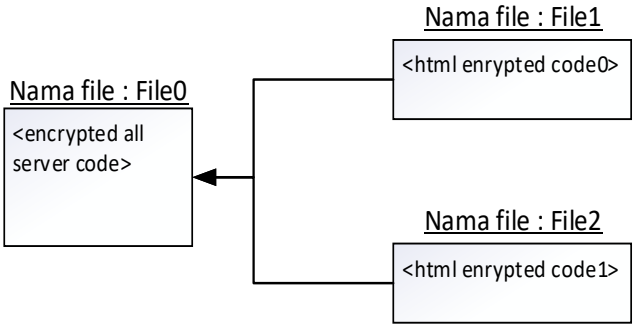

Fig. 5. File0, File1 dan File2 Terenkripsi

Sampai di sini proses enkripsi kode sumber baik dari sisi server maupun sisi client sudah teratasi dengan baik..

III. RESULTS AND DISCUSSIONS

Sebelum melakukan pengujian, terlebih dahulu dipersiapkan software yang digunakan meliputi: Sistem Operasi Windows 10, Database MySQL, Adobe ColdFusion Server, Adobe Dreamweaver Code Editor, dan Browser Firefox. Untuk kode sumber murni menggunakan ColdFusion Code sebagai kode server dan HTML Code sebagai kode client.

1. File Aplikasi Web Murni
   Kode sumber ini menggunakan nama file File0.cfm, berikut ini adalah listing kode sumber murninya.

```
<!--- server code0 --->
<cfset Nama = "Hasmira">
<cfset Alamat = "Malaysia">
<!--- html code0 --->
<html>
<head>
   <title>MENGATASI MASALAH KOMBINASI
   ENKRIPSI SERVER DAN CLIENT SIDE PADA
   KEAMANAN WEB CODE MENGGUNAKAN METODE
   SLICE CODE</title>
</head>
<body>
<!--- server code1 --->
<cfoutput>
   Nama ku adalah <b>#Nama#</b><br />
   Aku tinggal di <i>#Alamat#</i>
</cfoutput>
<hr />
<!--- server code2 --->
<cfset A = 1>
<cfset B = 2>
<cfset C = #A#+#B#>
<!--- html code1 --->
Mari kita belajar tentang penjumlahan.<br />
Nilai C adalah penjumlahan antara nilai A dan nilai B
<hr />
<!--- server code3 --->
<cfoutput>
B = # B # <br />
A = # A # <br />
C = # A #  +  # B # = #C#
</cfoutput>
```

</body>
</html>
Kode di atas berisi kode server yang dikombinasikan dengan kode html. Kode server melakukan inisialisasi terhadap variable Nama, Alamat, nilai A, nilai B, nilai C hasil penjumlahan, serta menampilkan nilai-nilai tersebut. Berikut ini adalah tampilan hasil dari kode di atas jika ditampilkan di browser web.

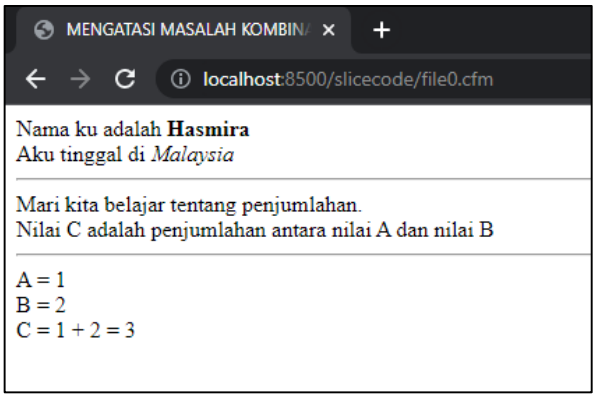

Fig. 6. Hasil Kode Sumber Asli

2. Pembagiaan -slice- kode sumber program menjadi file terpisah
Kode sumber sebelumnya yang berada pada file File0.cfm akan kita pisahkan menjadi 3 bagian -mengikuti alur metode- yaitu : 1 file server side dan 2 file client side. Berikut ini adalah hasil pemisahan kode sumber:
File0.cfm
<!--- server code0 --->
<cfset Nama = "Hasmira">
<cfset Alamat = "Malaysia">
<!--- server code1 --->
<cfoutput>
Nama ku adalah #Nama#
Aku tinggal di #Alamat#
</cfoutput>
<hr />
<!--- server code2 --->
<cfset A = 1>
<cfset B = 2>
<cfset C = #A#+#B#>
<!--- server code3 --->
<cfoutput>
B = # B #
A = # A #
C = # A # + # B # = #C#
</cfoutput>
File1.htm
<!--- html code0 --->
<html>
<head>
<title>MENGATASI MASALAH KOMBINASI ENKRIPSI SERVER DAN CLIENT SIDE PADA KEAMANAN WEB CODE MENGGUNAKAN METODE SLICE CODE</title>
</head>
<body>
File2.htm
<!--- html code1 --->
Mari kita belajar tentang penjumlahan.
Nilai C adalah penjumlahan antara nilai A dan nilai B
<hr />
Kode sumber di atas yang apabila dijalankan di browser hingga hendak menciptakan output semacam di dasar ini. Tampak jelas bahwa hasil yang ditampilkan berbeda dengan hasil sebelumnya, tapi tidak perlu khawatir karena ini sifatnya hanyalah sementara. Setelah nantinya dilakukan proses selanjutnya, maka hasil

akhir dari semua ini -setelah dienkripsi dan dikombinasikan- akan sama persis seperti yang ditampilkan pada bagian pertama, yaitu Aplikasi Web Murni.

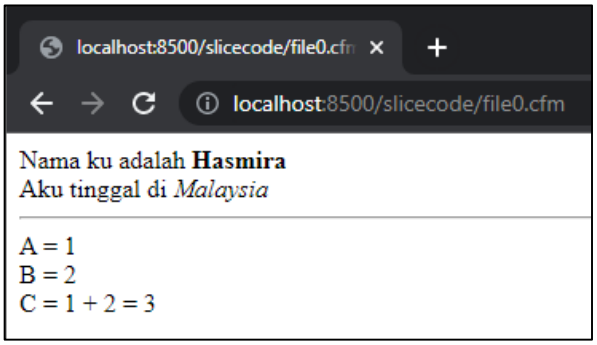

Fig. 7. Hasil Slice Code File0

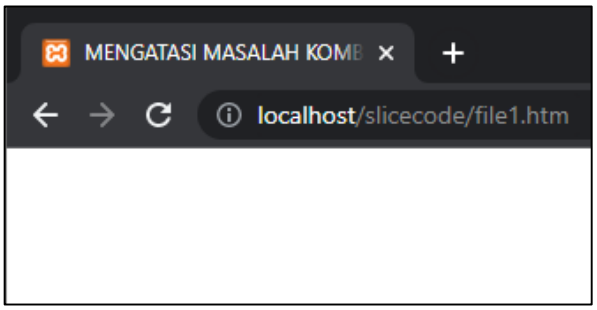

Fig. 8. Hasil Slice Code File1

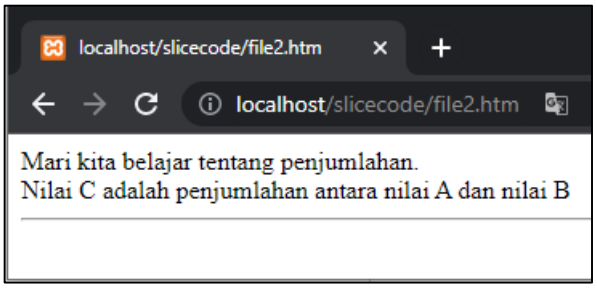

Fig. 9. Hasil Slice Code File2

Tampak pada gambar di atas tampilan web pada masing-masing halaman tidak menunjukkan pesan kesalahan -error-, ini artinya proses pembagian kode sumber telah bekerja dengan baik serta cocok dengan hasil yang diharapkan.

3. Mengenkripsi kode sumber client side yaitu 2 buah file html di atas agar kode sumber tidak bisa di tampilkan melalui browser web. Hasil dari enkripsi kode sumber html adalah seperti berikut ini:

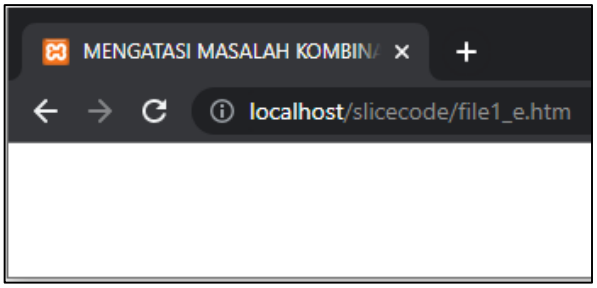

Fig. 10. Hasil Enkripsi File1

Hasil di atas merupakan tampilan dari file1.htm yang telah telah dienkripsi dan diganti nama filenya menjadi file1_e.htm, berikut ini adalah merupakan kode sumber hasil enkripsi:

Jurnal RESTIA Volume 1, Nomor 2, Agustus 2023 Hal 1-11
e-ISSN : 2988-5663

```
File1_e.htm
<script>
<!--
eval(unescape('%20%72%62%61%31%62%61%30%28%73%29%20%7b%0a%09%76%61%72%20%72
%20%3d%20%22%22%3b%0a%09%76%61%72%20%74%6d%70%20%3d%20%73%2e%73%70%6c%
69%74%28%22%3131%35%22%29%3b%0a%09%73%20%3d%20%75%6e%65%73%63%61%70%65
%28%74%6d%70%5b%30%5d%29%3b%0a%09%6b%20%3d%20%75%6e%65%73%63%61%70%65%
28%74%6d%70%5b%31%5d%20%2b%20%22%35%36%39%32%36%31%22%29%3b%0a%09%66%6f
%72%28%20%76%61%72%));
eval(unescape('%64%%63%75%2e%77%72%69%74%65%28%72%62%61%31%62%61%30%28%27')
+
'%39%26%2b%28%2f%23%6b%71%6b%65%22%65%6e%61%62%36%25%2f%2e%2e%3b%0b%03%
3e%6e%75%68%6b%38%08%08%3f%6b%60%67%6d%3c%0b%0b%39%73%6f%71%6e%66%3d%48
%43%47%45%47%55%44%54%4f%25%4f%42%50%44%4a%48%4a%26%4a%4a%4a%44%4c%4c%4
2%50%4c%26%4c%4c%4d%53%4c%57%55%4c%22%50%46%57%50%4c%50%26%45%44%49%26
%46%4e%4a%46%4b%52%29%51%4f%45%40%27%56%44%46%42%23%4e%43%48%4f%47%4f%4
4%49%26%52%47%41%23%46%49%4d%47%26%4c%40%49%41%42%57%4d%42%4e%47%47%22
%4b%44%51%48%42%40%22%50%4f%4c%45%4c%22%45%4e%41%42%3a%2a%76%6a%77%69%6
3%37%0f%0c%3d%2a%6f%63%64%66%3d%0e%0f%0b%03%3e%64%6e%61%7e%3811028215%35%
37%36%35%32%33%33' + unescape('%273b'));
// -->
</script>
<noscript</noscript>
</body>
</html>
```

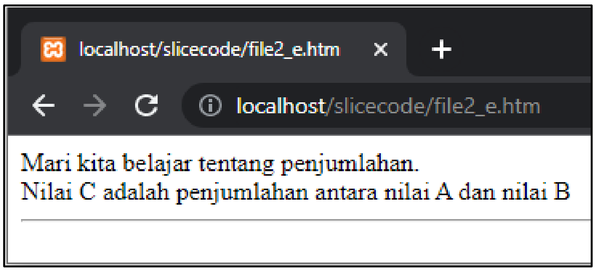

Fig. 11. Hasil Enkripsi File2_e.htm

Hasil di atas merupakan tampilan dari file2.htm yang telah telah dienkripsi dan diganti nama filenya menjadi file2_e.htm, berikut ini adalah merupakan kode sumber hasil enkripsi:

```
File2_e.htm
<script>
<!--
eval(unescape('%%6e%20%74%35%64%31%31%61%61%37%28%73%29%20%7b%0a%09%76%61%
72%20%72%20%3d%20%0a%09%76%61%72%20%74%6d%70%20%3d%20%73%2e%73%70%6c%69
%74%28%22%32%32%34%37%30%34%31%35%22%09%6b%20%3d%20%75%6e%65%73%63%61%
7028%74%6d%70%5b%31%5d%20%2b%20%22%36%37%30%36%36%35%22%29%3b%0a%09%66
%6f%72%28%20%76%61%%74%68%3b%20%69%2b%%7b%0a%09%09%72%20%2b%3d%20%53%
74%72%69%6e%67%2e%66%72%6f%6d%43%68%61%72%43%6f%64%65%28%28%70%61%72%73
%65%49%6e%74%28%6b%2e%63%68%61%72%41%74%28%69%25%6b%2e%6c%65%6e%67%74%
68%29%29%5e%73%2e%63%68%61%72%43%6f%64%65%41%74%28%69%29%29%2b%31%29%3b
%0a%09%7d%0a%09%72%65%74%75%72%6e%20%72%3b%0a%7d%0a'));
eval(unescape('%64%6f%%74%2e%77%72%69%74%65%28%74%35%64%31%31%61%61%37%28%2
7')                                                                    +
'%3f%21%2c%28%25%17%6e%75%6b%6b%19%64%6b%67%65%30%1b%25%24%25%3b%0b%09%
4a%66%74%6c%1e%6a%6c%7a%68%16%67%63%6b%66%6f%65%75%1e%73%60%64%7b%69%6b
%61%1f%69%62%68%6d%75%6c%6f%69%6f%69%6b%2a%3b%67%77%1a%2a%3c%0c%0d%44%6
0%62%66%6f%1f%44%19%65%67%61%6b%64%6e%17%66%62%6a%69%72%6a%6e%64%66%60%
69%16%68%64%75%67%71%66%19%68%6c%6a%60%6c%16%48%16%65%67%6d%19%6b%6d%6f
```

```
%61%68%1b%48%04%00%3d%60%71%19%28%3822470415%34%31%30%34%39%38%39' +
unescape('%27%29%29%3b'));
// -->
</script>
</body>
</html>
```

Daril hasil menjalankan 2 file html di atas, menunjukkan bahwa file yang telah terenkripsi dapat berjalan dengan baik tanpa menimbulkan efek yang berbeda dibandingkan dengan file html yang belum dienkripsi.

4. Menyisipkan file-file html yang telah dienkripsi ke dalam file server yang belum dienkripsi yaitu file0.cfm Berikut ini adalah kode sumber dari penyisipan file-file tersebut:

File0.cfm

```
<!--- server code0 --->
<cfset Nama = "Hasmira">
<cfset Alamat = "Malaysia">
<!--- html code0 --->
<cfinclude template="File1_e.htm">
<!--- server code1 --->
<cfoutput>
Nama ku adalah <b>#Nama#</b><br />
Aku tinggal di <i>#Alamat#</i>
</cfoutput>
<hr />
<!--- server code2 --->
<cfset A = 1>
<cfset B = 2>
<cfset C = #A#+#B#>
<!--- html code1 --->
<cfinclude template="File2_e.htm">
<!--- server code3 --->
<cfoutput>
B = # B # <br />A = # A # <br />
C = # A# + # B# = #C#
</cfoutput>
</body>
</html>
```

Setelah File0.cfm dijalankan pada browser web, maka hasil yang diberikan adalah seperti berikut ini:

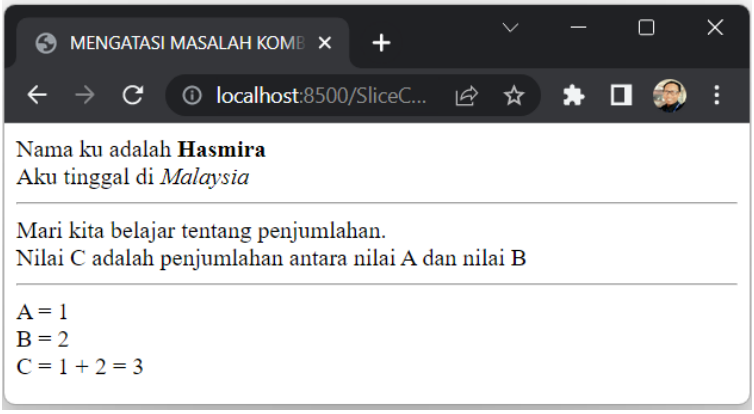

Fig. 12. Hasil Penyisipan File Kode Sumber Terenkripsi

Dari hasil di atas menunjukkan bahwa file server yang telah disisipkan oleh file-file html yang terenkripsi telah berjalan dengan baik di browser web tanpa menimbulkan pesan kesalahan apa pun.

5. Langkah terakhir adalah kita melakukan enkripsi terhadap file server. Jika Langkah ini terlaksana dengan baik, maka baik kode sumber server maupun kode sumber klien bisa terenkripsi dengan sempurna.

Untuk mengenkripsi kode sumber server ini menggunakan tools bawaan dari ColdFusion Environment yang telah tersedia. Berikut ini adalah kode sumber server yang telah terenkripsi:

File0_e.cfm
Oä
/'lë¹*â `À^FÕhqØ¤ä8X°É¿Ìò©‰P^qv
ßNÊÒ‡ùFÍû'ÉÊvà[®?á¥®ÑÓÌquÖ-Ô»àdN,û=ÒÂ“šüù>eÐ×åRÞ
Ñm¬èÔ{ý¡ • ¤²ãÁ?ÃÑ-ÿ‘ÑÐ•o-¾□#fD…‰Ñ□¤}ÔUkJÁ¡G¼yºã
Šóp€ö[b=l¿'3ÝèÔÕ5:qÂ›U·â}t¾Œªäݗ$k'iø ·?ÃÑ-ÿ‘ó5©Yv½y¥O
þþ{²Ò ±É‡Ç□fLôä0×‰t®PÝtï¼Ê‡5AC□8Œþë´FZ«‡7)È&¾bf¶-ÏèYÆ:6ò°û€}
Úé;»ÿú¦¢š¶tÊÖ9G0X|‹Á°I2-^ËÌb§
ˆ»ßêôítY-¦¯ñ02vAS□ãBÀŒ ú?N`C9V^XGãxž ÀfÁT[îžêFÔi ÎÝ?Q¾ñ(üŸôhë…Ák¾õ
MÎðë¯uJ&Ñ¯i
%i¤§7É‡¨°%¿×Ä¹P½(¥ • &e—NnŠW□ê¿□ÛJßnx sòš%Ã˜fÕ¬¯íÛÓ}
÷á¬E)ã˜™ƒ$sw-Ù=,N/ÔÞQ+p0E†ûHèË©çê²pCr‘…d:IÁœ–¥›ËB¼HØ&¤[ÑÔøRÇ-©•Õ¢®¦Ãp•
g |ÓÐ:²©ÑÑu˜È} ÕÃê^ËÌb§
ˆ? -è¢Éøõ^ËÌb§
ˆd¾□;¤ÜAYêóuQŸGÌLðmÀ/J)Þ¶-ÏèYÆ:68½Ñ{÷á°á¼ÈXšáì¥z®ï™k[@ jlhnnhl

Berikut ini adalah hasil akhir tampilan saat file File0_e.cfm dijalankan pada browser web,

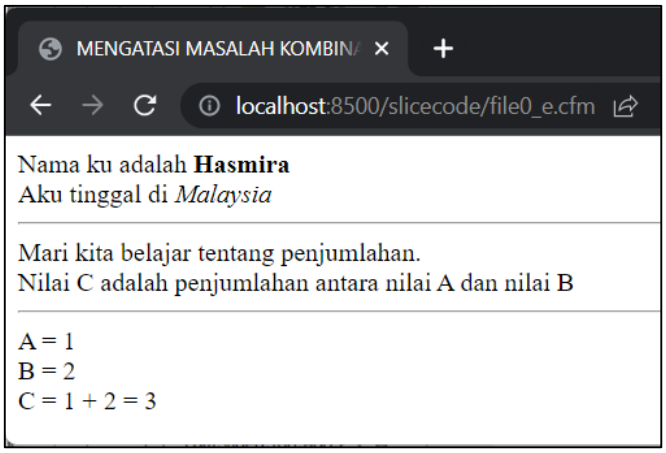

Fig. 13. Hasil Akhir Slice Code Terenkripsi

Dari tampilan di atas menunjukkan bahwa aplikasi web yang kode sumbernya telah terenkripsi dapat berjalan dengan baik tanpa memberikan pesan kesalahan apa pun.

IV. CONCLUSION

Berikut ini merupakan kesimpulan dari riset yang dilakukan:

1. Source code merupakan sumber daya aplikasi web yang rentan untuk disalahgunakan untuk kepentingan yang dapat merugikan.
2. Source code harus dilindungi dengan cara dienkripsi baik dari sisi server atau pun client.
3. Menggunakan satu metode enkripsi saja tidak cukup untuk melindungi source code dari kedua sisi.
4. Menggabungkan dua sisi metode enkripsi dengan menggunakan metode slice code, maka sangat dimungkinkan untuk mengamankan source code baik dari sisi server atau client.

References

[1] D. H. Curie, J. Jaison, J. Yadav, and J. R. Fiona, “Analysis on Web Frameworks,” *J. Phys. Conf. Ser.*, vol. 1362, no. 1, p. 12114, 2019, doi: 10.1088/1742-6596/1362/1/012114.

[2] A. S. Bhadouria, “Study of: Impact of Malicious Attacks and Data Breach on the Growth and Performance of the Company and Few of the World’s Biggest Data Breaches,” *Int. J. Sci. Res. Publ.*, 2022.

[3] K.-C. Chang, Y.-K. Gao, and S.-C. Lee, “The effect of data theft on a firm’s short-term and long-term market value,” *Mathematics*, vol. 8, no. 5, p. 808, 2020.

[4] A. N. Luthiya, B. Irawan, and R. Yulia, “Kebijakan Hukum Pidana Terhadap Pengaturan Pencurian

Data Pribadi Sebagai Penyalahgunaan Teknologi Komunikasi Dan Informasi," *J. Huk. Pidana dan Kriminologi*, vol. 2, no. 2, pp. 14–29, 2021.

[5] R. A. Prastyanti, I. Rahayu, E. Yafi, K. Wardiono, and A. Budiono, "Law And Personal Data: Offering Strategies For Consumer Protection In New Normal Situation In Indonesia," *J. Jurisprud.*, vol. 11, no. 1, pp. 82–99, 2022.

[6] D.-W. Huang, W. Liu, and J. Bi, "Data tampering attacks diagnosis in dynamic wireless sensor networks," *Comput. Commun.*, vol. 172, pp. 84–92, 2021.

[7] G. Ping, "Detection of Power Data Tampering Attack based on Gradient Boosting Decision Tree," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 12057.

[8] R. K. Shrivastava, S. Mishra, V. E. Archana, and C. Hota, "Preventing data tampering in IoT networks," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, IEEE, 2019, pp. 1–6.

[9] S. M. T. Situmeang, "Penyalahgunaan Data Pribadi Sebagai Bentuk Kejahatan Sempurna Dalam Perspektif Hukum Siber," *Sasi*, vol. 27, no. 1, pp. 38–52, 2021.

[10] V. N. N. Damanik and S. U. Sunaringtyas, "Secure code recommendation based on code review result using owasp code review guide," in *2020 International Workshop on Big Data and Information Security (IWBIS)*, IEEE, 2020, pp. 153–158.

[11] M. Aljabri *et al.*, "Testing and Exploiting Tools to Improve OWASP Top Ten Security Vulnerabilities Detection," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, 2022, pp. 797–803.

[12] A. Alanda, D. Satria, M. I. Ardhana, A. A. Dahlan, and H. A. Mooduto, "Web application penetration testing using SQL Injection attack," *JOIV Int. J. Informatics Vis.*, vol. 5, no. 3, pp. 320–326, 2021.

[13] C. Sharma, S. C. Jain, and A. K. Sharma, "Explorative study of SQL injection attacks and mechanisms to secure web application database-A," *Int J Adv Comput Sci Appl*, vol. 7, no. 3, pp. 79–87, 2016.

[14] M. O'Neill, "Insecurity by design: Today's IoT device security problem," *Engineering*, vol. 2, no. 1, pp. 48–49, 2016.

[15] B. Eshete, A. Villafiorita, and K. Weldemariam, "Early detection of security misconfiguration vulnerabilities in web applications," in *2011 Sixth International Conference on Availability, Reliability and Security*, IEEE, 2011, pp. 169–174.

[16] R. Poptani and M. V. Gatty, "Security Misconfiguration," *Secur. Misconfiguration*, vol. 7, no. 1, p. 3, 2018.

[17] E. K. Marino and A. J. Faas, "Is vulnerability an outdated concept? After subjects and spaces," *Ann. Anthropol. Pract.*, vol. 44, no. 1, pp. 33–46, 2020.

[18] R. Posso and S. Criollo-C, "Analysis of Authentication Failures in the Enterprise," in *New Knowledge in Information Systems and Technologies: Volume 2*, Springer, 2019, pp. 911–920.

[19] M. Zviran and Z. Erlich, "Identification and authentication: technology and implementation issues," *Commun. Assoc. Inf. Syst.*, vol. 17, no. 1, p. 4, 2006.

[20] D. V. S. Kaja, Y. Fatima, and A. B. Mailewa, "Data integrity attacks in cloud computing: A review of identifying and protecting techniques," *J. homepage www. ijrpr. com ISSN*, vol. 2582, p. 7421, 2022.

[21] J. Svacina *et al.*, "On vulnerability and security log analysis: A systematic literature review on recent trends," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2020, pp. 175–180.

[22] K. Al-talak and O. Abbass, "Detecting server-side request forgery (SSRF) attack by using deep learning techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 12, 2021.

[23] R. N. Ibrahim, "Kriptografi Algoritma Des, Aes/Rijndael, Blowfish Untuk Keamanan Citra Digital Dengan Menggunakan Metode Discrete Wavelet Transformation (Dwt)," *J. Comput. Bisnis*, vol. 6, no. 2, pp. 82–95, 2012.

[24] C. Riman and P. E. Abi-Char, "Comparative analysis of block cipher-based encryption algorithms: a survey," *Inf. Secur. Comput. Fraud*, vol. 3, no. 1, pp. 1–7, 2015.

[25] K. Curran, A. Bond, and G. Fisher, "HTML5 and the Mobile Web," *Int. J. Innov. Digit. Econ.*, vol. 3,

no. 2, pp. 40–56, 2012.

[26] M. H. Alfirdaus, M. Tahir, N. E. Dewanti, R. Ardianto, N. N. Azurah, and N. F. Cahyono, "Perancangan Aplikasi Enkripsi Deskripsi Mengunakan Metode Caesar Chiper Berbasis Web," *J. Tek. Mesin, Ind. Elektro dan Inform.*, vol. 2, no. 2, pp. 64–76, 2023.

[27] D. Adhar, "Implementasi Algoritma DES (Data Encryption Standard) Pada Enkripsi Dan Deskripsi SMS Berbasis Android," *JTIK (Jurnal Tek. Inform. Kaputama)*, vol. 3, no. 2, pp. 53–60, 2019.

[28] A. Elanda and R. L. Buana, "Analisis Keamanan Sistem Informasi Berbasis Website Dengan Metode Open Web Application Security Project (OWASP) Versi 4: Systematic Review," *CESS (Journal Comput. Eng. Syst. Sci.*, vol. 5, no. 2, pp. 185–191, 2020.