ISSN: 2477-5126
e-ISSN:2548-9356

Pengujian Keamanan Fitur *Upload File* Pada Sistem Aplikasi Web

Muhammad Anis Al Hilmi[1*)], Rahul Ken Yunan[2]
[1,2]Jurusan Teknik Informatika, Politeknik Negeri Indramayu
[1,2]Jalan Raya Lohbener Lama No. 8 Indramayu 45252
email: [1]alhilmi@polindra.ac.id, [2]rahulken96@gmail.com

***Abstract* – Web applications are one of the most commonly used platforms for sending information and services over the Internet. Even today, web applications are widely used for essential services. However, this massive use makes web applications a popular target for a series of threats in cyber attacks. The author applies secure coding by activating the file filter code on the example web application system. This research was conducted to understand how hackers take the opportunity of file upload vulnerability on web application systems. The goal is proof of concept secure coding when developing a web application. This study uses localhost with Apache2 web server on the device used for testing. The author prepares an example of a web application system with loopholes based on PHP and uploads it to the Github repository. To prove whether hackers can still get something from the web application system or not, the author conducted two test scenarios. In the first scenario, the author adds a .pdf file so that when the file is uploaded, it will return to the user to download the uploaded file in pdf form. Then, in the second scenario, the author applies secure coding to check the file extensions uploaded on the web application system. After the author conducted the design, discussion, and testing, it was concluded that the File upload vulnerability is a dangerous loophole for web applications because it can retrieve essential data. In addition, File upload vulnerability can be prevented by performing secure coding to filter uploaded files.**

Keywords : web, vulnerability, PHP, upload file, secure coding

***Abstrak* – Aplikasi *web* adalah salah satu *platform* yang paling umum digunakan untuk mengirim informasi dan layanan melalui Internet. Bahkan hingga saat ini aplikasi *web* pun banyak digunakan untuk layanan-layanan penting. Akan tetapi dengan masifnya penggunaan tersebut menjadikan aplikasi *web* sebagai target populer bagi serangkaian ancaman berupa serangan siber. Penulis menerapkan *secure coding* dengan cara mengaktifkan kode *filter file* pada contoh sistem aplikasi *web*. Penelitian ini dilakukan untuk memahami gambaran mengenai cara peretas mengambil kesempatan *file upload vulnerability* pada sistem aplikasi web. Tujuannya sebagai *proof of concept secure coding* pada saat mengembangkan sebuah aplikasi web. Pada penelitian ini menggunakan *localhost* dengan *web server* Apache2 pada perangkat yang digunakan untuk pengujian. Penulis menyiapkan contoh sistem aplikasi web yang mempunyai celah, berbasis PHP dan mengunggahnya pada *repository* Github. Untuk membuktikan apakah peretas masih dapat memperoleh sesuatu dari sistem aplikasi *web* atau tidak, penulis melakukan dua skenario pengujian. Pada skenario pertama, penulis menambahkan format .pdf, sehingga ketika file terunggah, akan kembali kepada user untuk mengunduh file yang telah diunggah dalam bentuk pdf. Lalu, skenario kedua, penulis menerapkan *secure coding* untuk mengecek ekstensi *file* yang diunggah pada sistem aplikasi *web* tersebut. Setelah penulis melakukan perancangan, pembahasan, dan pengujian, didapatkan kesimpulan bahwa *file upload vulnerability* merupakan suatu celah yang berbahaya bagi aplikasi *web* karena dapat mengambil data penting di dalamnya. Selain itu, *file upload vulnerability* dapat dicegah dengan melakukan *secure coding* untuk memfilter *file* yang diunggah.**

Kata Kunci – web*, *vulnerability*, PHP, *upload file*, *secure coding

*) **penulis korespondensi**: Muhammad Anis Al Hilmi
Email: alhilmi@polindra.ac.id

I. PENDAHULUAN

Aplikasi *web* adalah salah satu *platform* yang paling umum digunakan untuk mengirim informasi dan layanan melalui Internet. Bahkan hingga saat ini aplikasi *web* pun banyak digunakan untuk layanan-layanan penting. Akan tetapi dengan masifnya penggunaan tersebut menjadikan aplikasi *web* sebagai target populer bagi serangkaian ancaman berupa serangan siber [1]. Meskipun sebagian besar teknik telah dikembangkan untuk melindungi aplikasi *web* dan setidaknya meminimalisir serangan siber tersebut tetapi hanya ada sedikit upaya yang ditujukan untuk menarik koneksi di antara teknik-teknik ini dan membangun gambaran besar dari penelitian keamanan aplikasi *web*.

Kerentanan unggah *file* pada aplikasi *web* menjadi peluang bagi para peretas untuk mengunggah *file* dengan kode berbahaya yang selanjutnya dapat dieksekusi pada *server*. Setelah unggahan *file* dieksekusi para peretas dapat melakukan berbagai macam serangan siber seperti memasang perangkat di *web*, mencemari aplikasi *web*, menyebarkan *malware*, dan *phising*. Beberapa sisi *server* skrip (misalnya yang memiliki ekstensi seperti ".php", ".asp", dan ".js") memiliki kerentanan yang lebih signifikan sebab *server* tersebut diperlakukan sebagai *file* yang dapat dieksekusi secara otomatis dan tidak memerlukan izin sistem untuk dieksekusi [10]. *File Upload Vulnerability* adalah celah pada sistem keamanan saat mengunggah *file* pada suatu aplikasi/*website*. *File* yang diunggah, setelah dieksekusi, dapat diperlakukan dan dieksekusi secara otomatis, tidak memerlukan izin sistem *file* untuk eksekusi. Kerentanan unggahan *file* dianggap sebagai kerentanan *web* teratas oleh OWASP [4].

I.I. *Secure Coding*

ISSN: 2477-5126
e-ISSN:2548-9356

Secure Coding adalah metode penulisan perangkat lunak dan kode sumber yang terlindung dari serangan dunia maya. Dengan meningkatnya permintaan untuk proses *delivery* yang cepat, kadang membuat pelaku bisnis meninggalkan standar pengkodean yang aman/*best practice* dalam proses siklus hidup pengembangan perangkat lunak (SDLC) dan proses pengembangan mereka. Kelalaian seperti itu pada akhirnya akan menghilangkan manfaat jangka pendek dari distribusi cepat karena klien mulai menyuarakan adanya pelanggaran data mereka di media sosial dan di ruang publik [8].

I.II. RCE

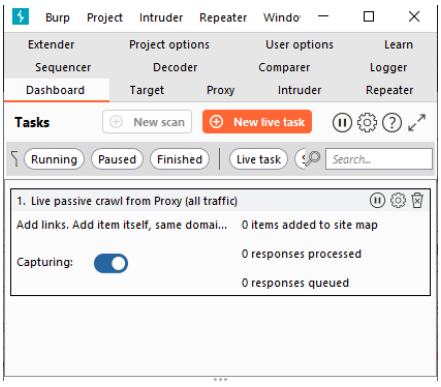
Remote Code Execution (RCE) adalah salah satu kerentanan serius di era ini. Menurut proyek *Web Application Security* (CWE / SANS), RCE telah terdaftar sebagai peringkat ke-2 aplikasi web kritis *Vulnerability* sejak 2016. RCE memungkinkan penyerang/*attacker* melakukan pengoperasian suatu perintah/skrip di suatu target dari jarak jauh [5]. RCE dapat terjadi setelah penyerang berhasil mengunggah *file* berbahaya ke suatu *server* dan menjalankan *file* tersebut untuk mengambil alih atau menjalankan serangkaian tindakan di luar kehendak sistem.

I.III. Burp Suite

Burp Suite adalah alat pengujian penetrasi yang digunakan oleh auditor keamanan, peneliti, dan penguji penetrasi. Pada Gambar 1, ditunjukkan logo *software* Burp Suite.

Gbr 1. Logo Burp Suite

Ini adalah *proxy* penyadap, oleh karena itu, Burp ditempatkan di antara server dan klien web sehingga semua permintaan dan tanggapan antara server dan klien web dirutekan dan ditangkap oleh Burp [12]. Pada Gambar 2, ditunjukkan tampilan dari *Software* Burp Suite.



Gbr 2. Contoh Tampilan *Software* Burp Suite

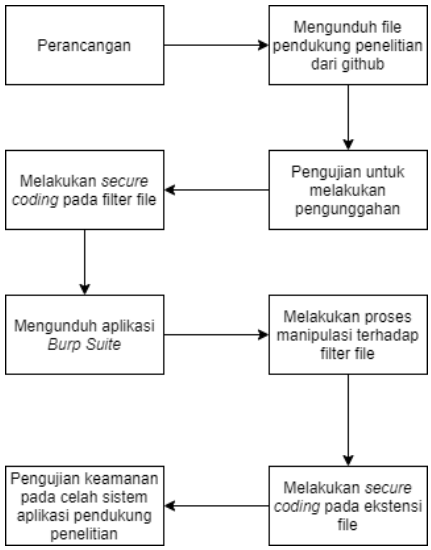
II. PENELITIAN YANG TERKAIT

Penelitian [2] menunjukkan adanya celah keamanan di sisi fitur *upload file*, yaitu tidak adanya pembatasan/*filter*, yaitu pada proyek *open-source* Oscommerce, semacam *content-management system* (CMS) yang sering digunakan toko *online*, digunakan mulai dari usaha skala kecil hingga besar, berbasis PHP dan MySQL. Celah tersebut ditangani dengan *patch*/menambal celah, dengan terlebih dahulu melakukan analisis kode program.

Pada [3] dipaparkan 16 skenario penyerangan yang menyasar celah keamanan pada fitur *upload* berbasis PHP. Di sana dijelaskan juga bagaimana mitigasinya. Ini untuk menunjukkan pentingnya perhatian atas keamanan kode program terutama yang terkait dengan masukan data dari pengguna, misalnya bagian unggah *file*. Artikel [7] memperkuat paparan [3], bahwa celah keamanan pada fitur *upload file* dapat menjadi jalan masuk bagi penyerang, terutama bila tidak ada filter atau validasi pada nama, isi, dan ukuran file yang diperbolehkan untuk diunggah.

III. METODE PENELITIAN

Dalam penelitian ini, akan ditunjukkan bagaimana bahayanya fitur *upload file* dengan validasi atau filter yang lemah, dan skenario penyerangan menggunakan *tool proxy* yang mencegat proses unggah data dengan terlebih dahulu melakukan manipulasi terhadap properti dari file, tujuannya agar lolos dari filter yang tidak terlalu kuat. Tahap Penelitian dapat dilihat pada Gambar 3.



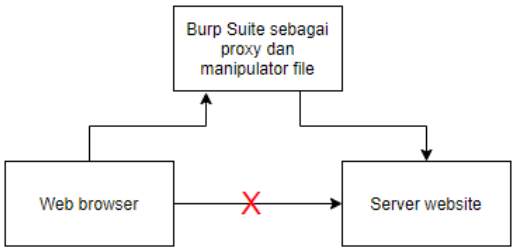
Gbr. 3 Tahap Penelitian

Penelitian ini diawali dari perancangan, dimana penulis mengunduh contoh *file* sistem aplikasi web yang memiliki *file upload vulnerability*. Setelah itu, penulis mencoba untuk melakukan pengunggahan *file*, lalu ditemukannya celah keamanan yang peretas dapat melihat isi data penting pada sistem aplikasi *web* tersebut. Kemudian penulis menerapkan *secure coding* dengan cara mengaktifkan kode *filter file* pada sistem tersebut.

Setelah itu, sistem sudah mulai aman, namun peretas tidak kehabisan akal untuk melakukan pencurian data dengan menggunakan bantuan dari aplikasi Burp Suite. Kemudian, implementasi dilakukan dengan menggunakan Burp Suite

ISSN: 2477-5126
e-ISSN:2548-9356

untuk proses memanipulasi *file* ilegal untuk dapat lolos pada saat pengunggahan *file* ke dalam sistem.

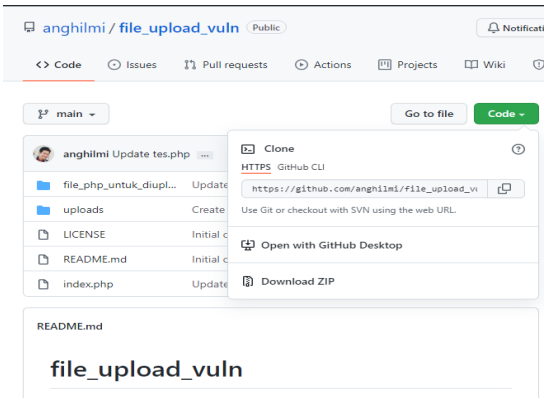


Gbr 4. Ilustrasi Pengujian

Di tahap akhir, dilakukan skenario pengujian keamanan dengan menerapkan *secure coding* pada *filter* ekstensi *file*, untuk mencegah *file* ilegal yang tidak diinginkan masuk ke dalam sistem aplikasi *web* tersebut.

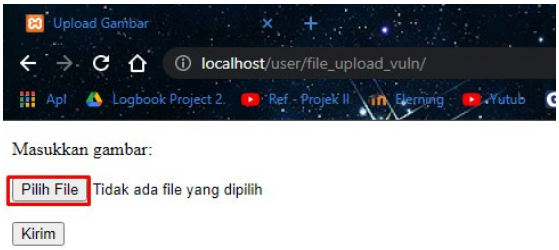
IV. HASIL DAN PEMBAHASAN

Penelitian ini dilakukan untuk memahami gambaran mengenai cara peretas mengambil kesempatan *file upload vulnerability* pada aplikasi *web*. Tujuannya sebagai *proof of concept secure coding* pada saat mengembangkan sebuah aplikasi *web*, juga untuk keperluan informasi & edukasi akan dampaknya pada sistem yang telah dibuat serta ditunjukkan pula cara mengatasi kejadian tersebut untuk kedepannya. Pada penelitian ini menggunakan *web server* Apache2 pada *localhost* pada perangkat yang digunakan untuk pengujian. Di sini penulis menyiapkan contoh sistem aplikasi *web* yang mempunyai celah dari situs alamat *website*, https://github.com/anghilmi/file_upload_vuln.



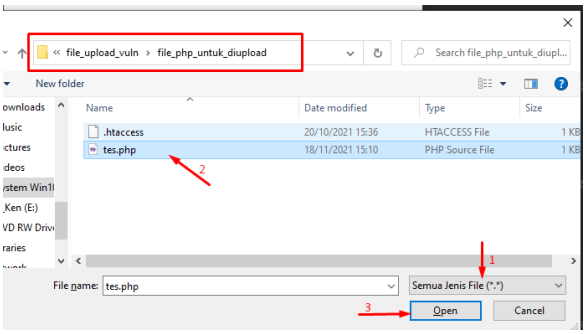
Gbr 5. Tangkapan Layar ketika akan mengunduh contoh sistem aplikasi *web*

Setelah mengakses alamat *website* yang tertera maka akan terbuka tampilan seperti pada Gambar 5. Terdapat tombol *code* > *Download zip* yang jika ditekan akan dapat mengunduh *file* zip tersebut. *File* zip yang telah diunduh diekstrak dan dimasukkan kedalam folder *xampp* > *htdocs* > dan dimasukkan ke folder baru bernama "user", lalu menjalankan sistem aplikasi web pada *localhost* dengan terlebih dahulu menjalankan *web server* Apache2 pada XAMPP, yang dapat dilihat pada Gambar 6.

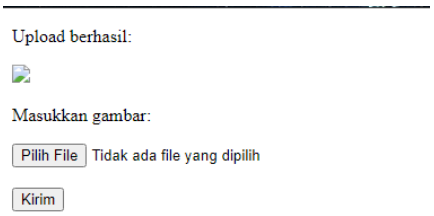


Gbr 6. Tangkapan Layar Contoh Sistem Aplikasi *Web*

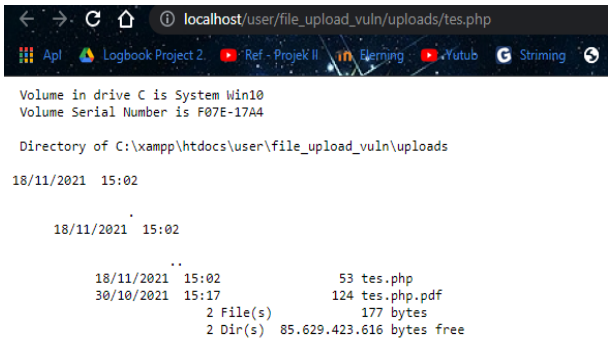
Pada Gambar 7 dan 8, penulis mengirim *file* .php untuk menguji celah pada sistem aplikasi *web* tersebut.



Gbr 7. Tangkapan Layar Unggah *File* php



Gbr 8. Tangkapan Layar Unggah Berhasil



Gbr 9. Tangkapan Layar Informasi Penting di Dalam Pada Sistem Aplikasi *Web*

Pada Gambar 9, terlihat bahwa ketika penulis mencoba untuk membuka *file* yang diunggah dalam bentuk gambar di tab lain, maka akan muncul informasi tentang *file* penting yang berada di folder yang dimana ini menjadi dasar dari *file upload vulnerability* pada sistem aplikasi *web* tersebut. Untuk mengatasinya penulis menerapkan *secure coding* dengan membuka folder *file_upload_vuln* > *index.php*.

ISSN: 2477-5126
e-ISSN:2548-9356

Pada Gambar 10.1 dan 10.2, penulis menerapkan *secure coding* dengan mengaktifkan kode *whitelist filter jenis file upload*.

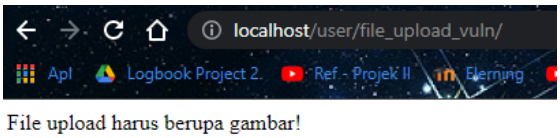
```
$whitelist_type = array('image/jpg',
'image/jpeg', 'image/png', 'image/bmp');
```

Gbr 10.1. Tangkapan Layar Kode *Filter Whitelist*

```
if (!in_array($_FILES['image']['type'], $whitelist_type)){
    die("File upload harus berupa gambar!");
}
```

Gbr 10.2. Tangkapan Layar Kode *Filter Whitelist*

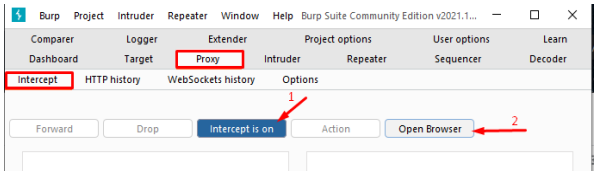
Lalu, penulis melakukan pengujian ulang untuk mengetes hasil penerapan *secure coding* pada sistem aplikasi *web* tersebut. Pada Gambar 11, terlihat bahwa penerapan *Secure Coding* tersebut berhasil memfilter file .php yang diunggah.



File upload harus berupa gambar!

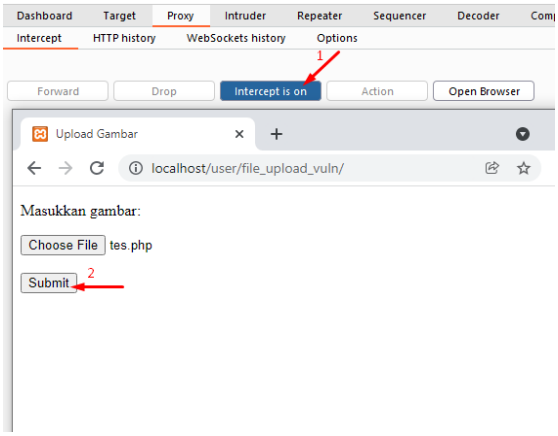
Gbr 11. Hasil dari *secure coding whitelist filter*

Tapi, belum sampai disini pasti para peretas tidak akan kehabisan akal untuk sampai mendapatkan sesuatu yang berharga dari suatu aplikasi *web*, peretas akan mencoba memanfaatkan aplikasi yang bisa membantu mereka untuk mengelabui sistem kode *filter* ini, contohnya dengan mengunduh aplikasi Burp Suite.



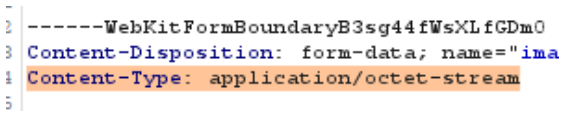
Gbr 12. Tangkapan Layar Halaman Tempat Pengujian Pada Burp Suite

Setelah aplikasi Burp Suite terbuka, pada bagian Proxy lalu *intercept*, maka akan terbuka tampilan seperti pada Gambar 12. Setelah itu penulis menekan tombol *open browser* untuk memunculkan jendela *browser* dalam keadaan *mode incognito*, kemudian menjalankan kembali *localhost* tempat folder sistem aplikasi web untuk penelitian. *http://localhost/user/file_upload_vuln/*, selanjutnya penulis kembali mengunggah *file* test.php yang akan digunakan untuk mengunggah *file* ke dalam sistem aplikasi *web*-nya, sebelum kirim pastikan *intercept* dalam keadaan *on*.



Gbr 13. Tangkapan Layar Saat Unggah *File*

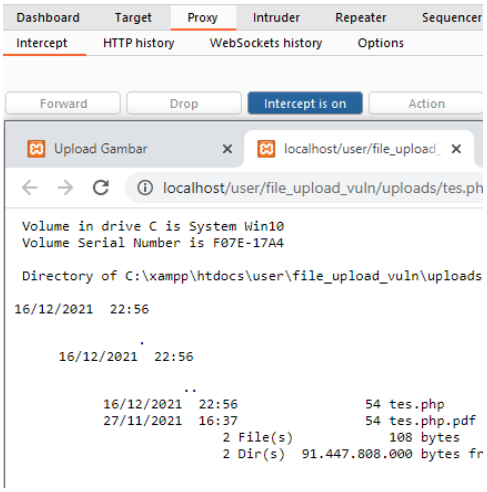
Ketika *file* test.php telah terunggah, proses pengunggahan akan terhenti oleh aplikasi Burp Suite untuk melihat detail dalam pengunggahan *file*, disini peretas dapat mengubah *Content-Type:application/octet-stream* menjadi *Content-Type : image/jpg*, yang tergambar pada Gambar 14.



Gbr 14. Tangkapan Layar Manipulasi File Saat Proses Pengunggahan

dari gambar tersebut merupakan langkah yang digunakan untuk mengelabui sistem *filter file* bahwa yang diunggah bukanlah sebuah gambar melainkan sebuah *file application* atau *file* ilegal lainnya yang seharusnya tidak diunggah ke dalam sistem tersebut.

Maka dengan manipulasi proses untuk mengelabui *filter* jenis *file* pada sistem tersebut. Pada Gambar 15, ditunjukkan bahwa penulis berhasil masuk untuk melihat *file* penting pada sistem tersebut.



Gbr 15. Tangkapan Latar *File* Penting Pada Sistem Aplikasi *Web*

ISSN: 2477-5126
e-ISSN:2548-9356

Untuk membuktikan apakah peretas masih dapat memperoleh sesuatu dari aplikasi *web* atau tidak, maka penulis memutuskan untuk menggunakan aplikasi Burp Suite untuk menguji ketahanan sistem aplikasi *web*, sekali lagi terhadap serangan peretas, penulis melakukan 2 skenario pengujian. Skenario pertama, penulis menambahkan format .pdf seperti tergambar pada Gambar 16.

```
. basename($_FILES['image']['name']) . ".pdf";
```

Gbr 16. Tangkapan Layar Kode Penambahan Format .pdf

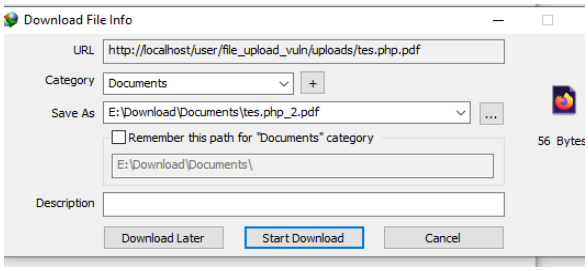
Agar ketika *file* terunggah, akan kembali kepada *user* untuk mengunduh *file* yang telah diunggah dalam bentuk pdf.

Lalu, skenario kedua, penulis menerapkan *secure coding* untuk mengecek ekstensi *file* yang diunggah pada sistem aplikasi *web* tersebut, yang digambarkan pada Gambar 17.

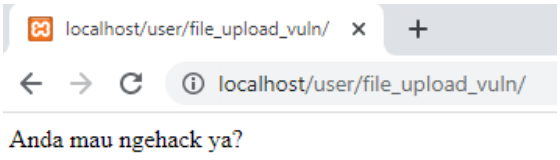
```
$cek_ekstensi = pathinfo($target_path, PATHINFO_EXTENSION);
if($cek_ekstensi == 'php'){
    die("Anda mau ngehack ya?");
}
```

Gbr 17. Tangkapan Layar Penerapan *Secure Coding* Cek Ekstensi File Unggah

Dan terbukti bahwa dari kedua skenario pengujian tersebut menunjukkan hasil pengujian untuk keamanan sistem aplikasi *web* berhasil, dan hasil seperti pengujian tergambar pada Gambar 18 dan 19 :



Gbr 18. Tangkapan Layar Hasil Skenario 1



Gbr 19. Tangkapan Layar Hasil Skenario 2

V. KESIMPULAN

Setelah penulis melakukan perancangan, implementasi, dan pengujian, dapat dihasilkan kesimpulan sebagai berikut:

1. *File upload vulnerability* merupakan suatu celah yang berbahaya bagi aplikasi web karena dapat mengambil data penting di dalamnya.
2. *File upload vulnerability* dapat dicegah dengan melakukan *secure coding* untuk memfilter *file* yang diunggah.
3. Dari pengujian, didapatkan bahwa peretas masih bisa mengelabui sistem *filter* dengan menggunakan aplikasi Burp Suite yang memanfaatkan pada proses dan memanipulasi jenis *file* yang diunggah. Dan ini pula dapat dicegah dengan melakukan *secure coding* untuk mengetahui ekstensi *file* yang diunggah.

DAFTAR PUSTAKA

Journal Article

[1] Li, X., & Xue, Y. (2011). A survey on web application security. Nashville, TN USA, 25(5), 1-14.

[2] Riadi, I., & Aristianto, E. I. (2016). An analysis of vulnerability web against attack unrestricted image file upload. Computer Engineering and Applications Journal, 5(1), 19-28.

[3] Pooj, K., & Patil, S. (2016). Understanding File Upload Security for Web Applications. International Journal of Engineering Trends and Technology, 42(7), 342-347.

Conference / Proceeding

[4] Huang, J., Li, Y., Zhang, J., & Dai, R. (2019, June). UChecker: Automatically detecting php-based unrestricted file upload vulnerabilities. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 581-592). IEEE.

[5] Biswas, Saikat, et al. "A study on remote code execution vulnerability in web applications." International Conference on Cyber Security and Computer Science (ICONCS 2018). 2018.

[6] O. Starov, J. Dahse, S. S. Ahmad, T. Holz, and N. Nikiforakis, "No honor among thieves: A large-scale analysis of malicious web shells," in Proceedings of the 25th International Conference on World Wide Web, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 1021–1032. [Online].

[7] Nagpure, S., & Kurkure, S. (2017, August). Vulnerability assessment and penetration testing of Web application. In 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA) (pp. 1-6). IEEE.

Book

[8] Rao, U. H., & Nayak, U. (2014). The InfoSec handbook: An introduction to information security (p. 392). Springer Nature.

[9] Solichin, A. (2016). Pemrograman web dengan PHP dan MySQL. Penerbit Budi Luhur.

[10] Hilmi, Muhammad Anis Al. 2021. Superlab Cybersecurity : Pengantar Keamanan Komputer dengan Praktikum. Bandung: Manggu Makmur Tanjung Lestari. ISBN: 9786236003336

ISSN: 2477-5126
e-ISSN:2548-9356

[11] Mearaj, I., Maheshwari, P., & Kaur, M. J. (2018, November). Data conversion from traditional relational database to MongoDB using XAMPP and NoSQL. In 2018 Fifth HCT Information Technology Trends (ITT) (pp. 94-98). IEEE.
[12] Kim, J. (2020). Burp suite: Automating web vulnerability scanning (Doctoral dissertation, Utica College).

Information from *i*nternet
[13] Williams, Alex (9 July 2012). "GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz". TechCrunch. Andreessen Horowitz is investing an eye-popping $100 million into GitHub.
[14] Yuliano, T. (2007). Pengenalan Php. IlmuKomputer. com.
[15] Borke, L., & Härdle, W. K. (2017). GitHub API based QuantNet Mining infrastructure in R. Available at SSRN 2927901.