BAB III

TINJAUAN PUSTAKA

3.1 Teori Pendukung

3.1.1 HTTP Header

Menurut Medly (2018) *Header* HTTP memungkinkan klien dan *server* menyampaikan informasi tambahan dengan permintaan atau tanggapannya. *Header* permintaan terdiri dari nama *case-insensitive* yang diikuti oleh titik dua ':', kemudian nilainya (tanpa jeda baris).

Header eksklusif dapat ditambahkan dengan menggunakan awalan 'X-', namun konvensi ini tidak berlaku lagi pada bulan Juni 2012, karena ketidaknyamanan yang diakibatkannya ketika bidang non-standar menjadi standar di RFC 6648; yang lain terdaftar di registri IANA, yang konten aslinya didefinisikan di RFC 4229. IANA juga mengelola sebuah registri dari *header* pesan HTTP baru yang diusulkan. *Header* dapat dikelompokkan menurut konteksnya:

a. *Header* umum: *Header* mengajukan permintaan dan tanggapan namun tidak ada hubungannya dengan data yang akhirnya ditransmisikan ke dalam tubuh.
b. Minta *header*: *Header* berisi lebih banyak informasi tentang sumber daya yang akan diambil atau tentang klien itu sendiri.
c. *Header respon*: *Header* dengan informasi tambahan tentang respon, seperti lokasinya atau tentang *server* itu sendiri (nama dan versinya dll

d. *Entity header*: *Header* berisi lebih banyak informasi tentang badan entitas, seperti panjang isinya atau tipe MIME-nya.

Header juga dapat dikelompokkan menurut bagaimana proxy menangani mereka:

a. *Header end-to-end*

Header ini harus dikirim ke penerima akhir pesan; yaitu, *server* untuk permintaan atau klien untuk sebuah respon. *Proxy* perantara harus mentransmisikan *header end*-to-*end* yang tidak dimodifikasi dan *cache* harus menyimpannya.

b. *Hop-by-hop header*

Header ini hanya bermakna untuk koneksi tingkat *transport* tunggal dan tidak boleh dipancarkan ulang oleh *proxy* atau *cache*. *Header* seperti itu adalah: *Connection*, *Keep-Alive*, *Proxy-Authenticate*, *Proxy-Authorization*, *TE*, *Trailer*, *Transfer-Encoding* dan *Upgrade*. Perhatikan bahwa hanya *header hop-by-hop* yang dapat disetel menggunakan header umum Sambungan.

Field HTTP *Header Request* dapat dilihat pada tabel 3.1 dan *field* HTTP *Header Respone* dapat dilihat pada tabel 3.2.

Tabel 3.1 HTTP *Request Field*

Nama *Header* Field	Deskripsi
Accept	*Content-Types that are acceptable for the response. See Content negotiation.*
Accept-Charset	*Character sets that are acceptable.*

Accept-Encoding	*List of acceptable encodings. See HTTP compression.*
Accept-Language	*List of acceptable human languages for response. See Content negotiation.*
Accept-Datetime	*Acceptable version in time.*
Access-Control-Request-Method, Access-Control-Request-Headers	*Initiates a request for cross-origin resource sharing with Origin (below).*
Authorization	*Authentication credentials for HTTP authentication.*
Cache-Control	*Used to specify directives that must be obeyed by all caching mechanisms along the request-response chain.*
Connection	*Control options for the current connection and list of hop-by-hop request fields. Must not be used with HTTP/2.*
Cookie	*An HTTP cookie previously sent by the server with Set-Cookie (below).*
Content-Length	*The length of the request body in octets (8-bit bytes).*
Content-MD5	*A Base64-encoded binary MD5 sum of the content of the request body.*
Content-Type	*The MIME type of the body of the request (used with POST and PUT requests).*
Date	*The date and time that the message was originated (in "HTTP-date" format as defined by RFC 7231 Date/Time Formats).*
Expect	*Indicates that particular server behaviors are required by the client.*
Forwarded	*Disclose original information of a client connecting to a web server through an HTTP proxy.*
From	*The email address of the user making the request.*
Host	*The domain name of the server (for virtual hosting), and the TCP port number on which the server is listening. The port number may be*

	omitted if the port is the standard port for the service requested. *Mandatory since HTTP/1.1. If the request is generated directly in HTTP/2, it should not be used.*
If-Match	*Only perform the action if the client supplied entity matches the same entity on the server. This is mainly for methods like PUT to only update a resource if it has not been modified since the user last updated it.*
If-Modified-Since	*Allows a 304 Not Modified to be returned if content is unchanged.*
If-None-Match	*Allows a 304 Not Modified to be returned if content is unchanged, see HTTP ETag.*
If-Range	*If the entity is unchanged, send me the part(s) that I am missing; otherwise, send me the entire new entity.*
If-Unmodified-Since	*Only send the response if the entity has not been modified since a specific time.*
Max-Forwards	*Limit the number of times the message can be forwarded through proxies or gateways.*
Origin	*Initiates a request for cross-origin resource sharing (asks server for Access-Control-* response fields).*
Pragma	*Implementation-specific fields that may have various effects anywhere along the request-response chain.*
Proxy-Authorization	*Authorization credentials for connecting to a proxy.*
Range	*Request only part of an entity. Bytes are numbered from 0. See Byte serving.*
Referer [sic]	*This is the address of the previous web page from which a link to the currently requested page was followed. (The word "referrer" has been misspelled in the RFC as well as in most implementations to the point that it has become standard usage and is considered correct terminology)*
TE	*The transfer encodings the user agent is willing to accept: the same values as for the response header field Transfer-Encoding can be used, plus*

	the "trailers" value (related to the "chunked" transfer method) to notify the server it expects to receive additional fields in the trailer after the last, zero-sized, chunk. *Only trailers is supported in HTTP/2.*
User-Agent	*The user agent string of the user agent.*
Upgrade	*Ask the server to upgrade to another protocol. Must not be used to upgrade to HTTP/2.*
Via	*Informs the server of proxies through which the request was sent.*
Warning	*A general warning about possible problems with the entity body.*

Sumber: Owasp.org

Tabel 3.2 HTTP Respone Field

Nama *Header* Field	Deskripsi
Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Access-Control-Expose-Headers, Access-Control-Max-Age, Access-Control-Allow-Methods, Access-Control-Allow-Headers	*Specifying which web sites can participate in cross-origin resource sharing*
Accept-Patch	*Specifies which patch document formats this server supports*
Accept-Ranges	*What partial content range types this server supports via byte serving*
Age	*The age the object has been in a proxy cache in seconds*
Allow	*Valid methods for a specified resource. To be used for a 405 Method not allowed*
Alt-Svc	*A server uses "Alt-Svc" header (meaning Alternative Services) to indicate that its resources can also be accessed at a different*

	network location (host or port) or using a different protocol *When using HTTP/2, servers should instead send an ALTSVC frame.*
Cache-Control	*Tells all caching mechanisms from server to client whether they may cache this object. It is measured in seconds*
Connection	*Control options for the current connection and list of hop-by-hop response fields.* *Must not be used with HTTP/2.*
Content-Disposition	*An opportunity to raise a "File Download" dialogue box for a known MIME type with binary format or suggest a filename for dynamic content. Quotes are necessary with special characters.*
Content-Encoding	*The type of encoding used on the data. See HTTP compression.*
Content-Language	*The natural language or languages of the intended audience for the enclosed content*[37]
Content-Length	*The length of the response body in octets (8-bit bytes)*
Content-Location	*An alternate location for the returned data*
Content-MD5	*A Base64-encoded binary MD5 sum of the content of the response*
Content-Range	*Where in a full body message this partial message belongs*
Content-Type	*The MIME type of this content*
Date	*The date and time that the message was sent (in "HTTP-date" format as defined by RFC 7231)*
ETag	*An identifier for a specific version of a resource, often a message digest*
Expires	*Gives the date/time after which the response is considered stale (in "HTTP-date" format as defined by RFC 7231)*

Last-Modified	*The last modified date for the requested object (in "HTTP-date" format as defined by RFC 7231)*
Link	*Used to express a typed relationship with another resource, where the relation type is defined by RFC 5988*
Location	*Used in redirection, or when a new resource has been created.*
P3P	*This field is supposed to set P3P policy, in the form of `P3P:CP="your_compact_policy"`. However, P3P did not take off, most browsers have never fully implemented it, a lot of websites set this field with fake policy text.*
Pragma	*Implementation-specific fields that may have various effects anywhere along the request-response chain.*
Proxy-Authenticate	*Request authentication to access the proxy.*
Public-Key-Pins	*HTTP Public Key Pinning, announces hash of website's authentic TLS certificate*
Retry-After	*If an entity is temporarily unavailable, this instructs the client to try again later. Value could be a specified period of time (in seconds) or a HTTP-date.[42]*
Server	*A name for the server*
Set-Cookie	*An HTTP cookie*
Strict-Transport-Security	*A HSTS Policy informing the HTTP client how long to cache the HTTPS only policy and whether this applies to subdomains.*
Trailer	*The Trailer general field value indicates that the given set of header fields is present in the trailer of a message encoded with chunked transfer coding.*
Transfer-Encoding	*The form of encoding used to safely transfer the entity to the user. Currently defined methods are: chunked, compress, deflate, gzip, identity.*

	Must not be used with HTTP/2.
Tk	*Tracking Status header, value suggested to be sent in response to a DNT(do-not-track).*
Upgrade	*Ask the client to upgrade to another protocol. Must not be used to upgrade to HTTP/2*
Vary	*Tells downstream proxies how to match future request headers to decide whether the cached response can be used rather than requesting a fresh one from the origin server.*
Via	*Informs the client of proxies through which the response was sent.*
Warning	*A general warning about possible problems with the entity body.*
WWW-Authenticate	*Indicates the authentication scheme that should be used to access the requested entity.*
X-Frame-Options	*Clickjacking protection: deny - no rendering within a frame, sameorigin - no rendering if origin mismatch, allow-from - allow from specified location, allowall - non-standard, allow from any location*

Sumber: Owasp.org

3.1.2 Keamanan Sistem

Masalah keamanan merupakan salah satu aspek penting dari sebuah sistem Informasi, sayang sekali masalah keamanan ini seringkali kurang dapat perhatiaan dari pemilik dan pengelola sistem informasi. Jatuhnya informasi ke pihak lain (misal pihak lawan bisnis) dapat menimbulkan kerugian bagi pemilik informasi. Untuk itu keamanan dari sistem informasi yang digunakan harus terjamin dalam batas yang diterima (Raharjo, 2012).

Keamanan informasi adalah penjagaan informasi dari sebuah ancaman yang mungkin terjadi dalam upaya untuk memastikan atau menjamin

kelangsungan bisnis (*business continuity*), meminimalisir resiko bisnis (*reduce business risk*) dan memaksimalkan atau mempercepat pengambilan investasi dan peluang bisnis (Sarno dan Iffano, 2009:27).

3.1.3 Serangan-serangan pada Keamanan *Website*

Setiap orang dapat menjadi penyerang dan menembus suatu aplikasi berbasis *web*. Penyerang tersebut dalam menyerang situs *web* tersebut memiliki motif. Motif yang di jalankan oleh penyerang pun bermacam-macam, ada yang hanya iseng, ada yang sekedar ingin tahu konten *web* tersebut, ada yang hanya ingin menguji kemampuan, namun banyak juga yang motifnya tidak baik seperti, Ingin mengambil data-data *web* tersebut untuk kepentingan pribadi dan bahkan ingin mengubah dan merusak isi *web* tersebut (Yudistira, 2012:13).

3.1.4 Macam-macam pengelompokan *hacker*

Menurut S'to (2009), penyerang atau *hacker* bisa dikelompokan berdasarkan aktifitas yang mereka lakukan, yaitu :

a. Black Hat Hacker

Black Hat Hacker adalah jenis *hacker* yang menggunakan kemampuan mereka untuk melakukan hal-hal yang dianggap melanggar hukum dan merusak.

b. White Hat Hacker

White Hat Hacker adalah jenis *hacker* yang menggunakan kemampuan mereka untuk menghadapi *Black Hat Hacker*.

c. *Grey Hat Hacker*

Grey Hat Hacker adalah jenis *hacker* yang bergerak diwilayah abu-abu, terkadang mereka adalah *White Hat Hacker*.

d. *Suicide Hacker*

Suicide Hacker merupakan *hacker* yang tidak takut dengan ancaman hukum sekalipun dan hanya mempunyai tujuan membuat kekacauan yang sebesar-besarnya.

3.1.5 *Vulnerability Assessment*

Menurut Siagian, Akbar, dan Andri (2013), *Vulnerability* atau celah keamanan adalah suatu kelemahan yang mengancam nilai *integrity*, *confidentiality* dan *availability* dari suatu aset. *Vulnerability* tidak hanya berupa *software bugs* atau kelemahan *security* jaringan. Namun kelemahan seperti pegawai yang tidak ditraining, dokumentasi yang tidak tersedia maupun prosedur yang tidak dijalankan dengan benar. *Vulnerability* bisa dikategorikan kedalam tiga bagian, yaitu kelemahan pada *system* itu sendiri, jalur akses menuju kelemahan sistem, serta kemampuan dari seorang hacker untuk melakukan *attacking.*

Menurut Goel dan Mehtre (2015), *Vulnerability assessment* (VA) adalah proses pemindaian sistem atau perangkat lunak atau jaringan untuk mengetahui kelemahan dan celah dalam hal itu. Ini celah bisa memberikan *backdoor* ke penyerang untuk menyerang korban. Sebuah sistem mungkin memiliki *access control vulnerability, boundary condition vulnerability, input validation*

vulnerability, authentication vulnerabilities, configuration weakness vulnerabilities, dan *exception handling vulnerabilities*.

Proses VA sendiri memiliki berbagai tingkatan, sehingga perusahaan dapat memilih tingkatan mana yang akan digunakan dalam pengukurannya. Tingkatan ini dapat disesuaikan dengan kondisi di masing-masing tempat.

a. Tingkat I : Pengukuran peraturan dan kebijakan (*policy assessment*). Pengukuran ini meliputi peraturan, kebijaksanaan, standar operasi di *client* dalam cakupan keamanan informasi.
b. Tingkat II : Evaluasi Jaringan

   Pengukuran ini meliputi kinerja jaringan, keamanan jaringan hingga ancaman-ancaman terhadap jaringan kerja. Pengukuran jenis ini memerlukan alat bantu seperti *scanning* atau *data capture.* Evaluasi jaringan ini bertujuan untuk mendapatkan informasi mengenai kondisi sebenarnya yang terjadi dilapangan, bagaimana tingkat kesadaran akan keamanan informasi yang sudah diterapkan selama ini.
c. Tingkat III : Test Penetrasi (*Penetration Testing*)

   Penetration testing sebenarnya menggunakan prinsip yang sama dengan *network evaluation* dimana pembedanya bahwa *Penetration testing* dilakukan dalam kondisi gelap, tanpa mengetahui konfigurasi dan kondisi sebenarnya seperti apa. Pada tes penetrasi maka *assessor* akan menjumpai *system* sebagai sebuah kotak tertutup menghadapi penetrasi yang data dari luar.

Kegiatan *Vulnerability Assessment* ini sangat dianjurkan untuk dilakukan secara rutin. Bisa dilakukan perminggu atau perbulan. Hal ini dikarenakan *trend* ancaman atau serangan selalu berkembang. Mulailah sedini mungkin untuk *aware* melakukan hal-hal kecil yang bisa menjaga keamanan *system* informasi kita karena satu hal yang pasti adalah tidak ada satupun yang aman di dunia maya. (GOV-CSIRT, 2012)

3.1.6 *Open Web Application Security Project* (*OWASP*)

Open Web Application Security Project (*OWASP*) adalah sebuah organisasi internasional yang bersifat *non-profit*, didirikan oleh *OWASP foundation* pada 21 April 2004 di Ameriaka Serikat. *OWASP* fokus pada peningkatan keamanan perangkat lunak dan didedikasikan untuk memungkinkan organisasi dalam mengembangkan, memperoleh, mengoperasikan, dan memelihara aplikasi terpercaya untuk menjamin keamanan yang dibuat atau dikembangkan. *OWASP* memiliki misi untuk mengamankan *software*, sehingga orang-orang dan organisasi dapat membuat keputusan terhadap resiko keamanan yang benar.

OWASP merupakan *vendor* netral yang tidak berafiliasi dengan perusahaan teknologi manapun, tidak mendukung atau merekomendasikan produk atau layanan komersial. Proyek yang sudah dibuat dan dipublikasikan ada 363 proyek dan semua berkaitan dengan keamanan aplikasi.

3.1.7 *Penetration Testing*

Menurut Ali dan Heriyanto (2011:39-40), menjelaskan bahwa *penetration testing* adalah proses melakukan eksplorasi dan eksploitasi lebih

jauh ke dalam sistem target setelah mengetahui *bug* atau *security hole* yang di *report* pada proses *vulnerability assesment*. Pada proses ini aktivitas *gainin access* coba dilakukan oleh *attacker*. *Attacker* akan mencoba mengeksploitasi celah keamanan sehingga berhasil mendapatkan akses ke dalam sistem target. Aktivitas *penetration testing* bisa dikatakan sebagai aktivitas *gaining access* yang lebih spesifik dibandingkan dengan *vulnerability assessment*, karena pada proses ini *attacker* sudah terlebih dahulu mengetahui kondisi target. Selain itu pada proses *penetration testing* sudah tidak ada lagi *false positive* karena *attacker* sudah mengetahui dengan benar kondisi target. Setelah melakukan analisa kerentanan sistem, pada proses ini peneliti berusaha untuk mendapatkan akses pada mesin target dengan melakukan *exploitasi*.

3.1.8 Macam-macam bentuk serangan pada keamanan *website*

Bentuk-bentuk serangan terhadap *website* menurut (Zam, 2011) :

1. ***Local File Inclusion* (LFI) &*Remote File Inclusion* (RFI)**

   Local File Inclusion (LFI) adalah sebuah lubang pada sebuah situs *web* sehingga memungkinkan seseorang bisa mengakses semua *file* di dalam *serve*r dengan hanya melalui *URL*. Sedangkan *Remote File Inclusion* (RFI) adalah sebuah lubang dimana sebuah situs mengizinkan seseorang meng-*include*-kan *file* dari luar *server*.

2. ***Cross-Site Scripting* (XSS)**

   Terkadang, dari pada menyerang sebuah *server* yang lebih sulit, seorang *hacker* bisa saja memanfaatkan kelemahan yang ada pada sisi *client*. Lagipula, aksi *hacking* yang satu ini agak sulit dideteksi karena

bekerja dari sisi *client*. *Cross-Site Scripting* yang disingkat XSS, bukan CSS, karena CSS digunakan untuk istilah *Cascarding Style Sheets* yang merupakan salah satu bahasa pemrograman *web* untuk mengendalikan beberapa komponen dalam sebuah situs *web* sehingga akan lebih terstruktur dan seragam yang dipakai untuk memformat tampilan *web* yang dibuat dengan bahasa *HTML* dan *XHTML*.

Dengan XSS, serangan dilakukan dengan cara menginjeksi/memasukan *script* ke dalam situs *web* melalui sebuah *browser*. Aksi XSS ini adalah dengan memanfaatkan metode *HTTP* (*Hypertext Transfer Protocol*) *GET* atau *HTTP POST*. Contoh paling sederhana yang menjadi target sasaran adalah buku tamu (*guest book*). Apabila terdapat buku tamu di sebuah situs, coba di isi dengan :

```
<script language=”javascript”>window.alert (‘I love this site’)</script>
```

Dibebaskan mengganti kata-kata ‘*I love this site’* dengan kata-kata personal. Jika keluar *alert javascript* di *browser* berarti situs tersebut bisa di XSS.

3. *Phising*

Pada teknik *hacking* yang satu ini, tekniknya adalah berusaha membuat seseorang mengunjungi situs yang salah sehingga memberikan informasi rahasia berupa *username* dan *password* maupun hal lainnya. Umumnya pelaku membuat situs yang memiliki nama *domain* mirip dengan aslinya, Istilah *phising* identik dengan *web spoofing* dan *DNS*

spoofing. Teknik *phising* ini juga dikenal dengan sebutan *fake login*, dimana seorang *login* dihalaman yang bukan sebenarnya.

4. SQL *Injection*

SQL *injection* merupakan sebuah aksi *hacking* yang dilakukaan di aplikasi *client* dengan cara memodifikasi printah atau *syntax* SQL. Terdapat dua jenis SQL *injection* :

1. *Blind* SQL *Injection,* teknik ini dilakukan dengan memasukan *syntax*/perintah SQL pada sebuah *web* yang memiliki *vulnerability* (kerentanan/celah keamanan) untuk melihat isi dari *database* SQL tersebut.
2. *Advance* SQL *Injection*, merupakan teknik tingkat lanjut buat SQL *Injection*, dimana tidak hanya bisa mengakses *database*, tetapi bisa membuat *shell* atau pun *backdoor* pada *site* target. Pada teknik ini, *hacker* akan mencoba melakukan serangan SQL *Injection* dan memasang *shell* di dalam situs menggunakan *syntax* SQL.

5. *Defacing*

Defacing adalah kegiatan *hacking* yang mengubah tampilan sebuah situs *web*. Sedangkan cara yang digunakan bisa bermacam-macam seperti SQL *Injection*, mencari *password*, dan cara lainnya.

6. MITM (*Man In The Middle*) *Attack*

Menurut Shubh dan Sharma (2016), Serangan *man-in-the-middle* adalah serangan dimana penyerang diam-diam melakukan *relay* dan mungkin mengubah komunikasi antara dua pihak yang percaya mereka

saling berkomunikasi satu sama lain. Serangan ini memungkinkan si penyerang untuk mencuri data seperti *username* dan *password* hanya dengan menjadi penengah antara *client* dan *server*.

7. *Session Hijacking*

Menurut Fauziah (2009), *Session hijacking* merupakan aksi pengambilan kendali *session* milik *user* lain setelah sebelumnya "pembajak" berhasil memperoleh autentifikasi *ID session* yang biasanya tersimpan dalam *cookies*. *Session hijacking* menggunakan metode *captured, brute forced* atau *reserve enggineered* guna memperoleh *ID session,* yang untuk selanjutnya pembajak memegang kendali atas *session* yang dimiliki oleh *user* lain tersebut selama *session* berlangsung

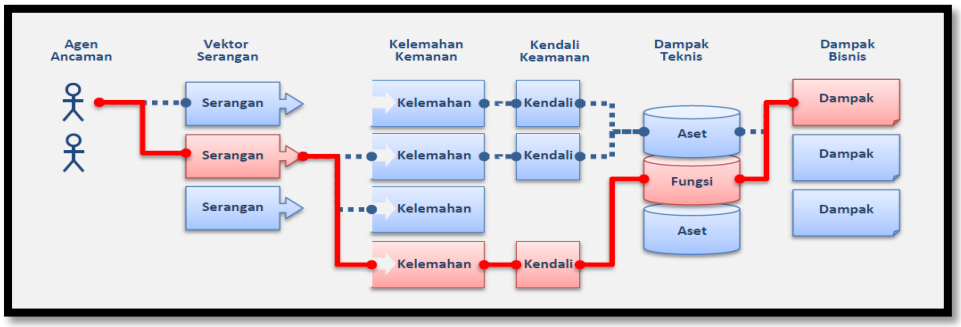
8. DDOS (*Distributed Denial of Service*)

Menurut Habibi, Rendy, dan Yovita (2015), DDoS adalah salah satu tipe dari *cyberattack* yang paling rumit, dimana mendapat perhatian besar pada jaringan di pemerintahan dan institusi–institusi besar saat ini. Serangan ini bersifat tersembunyi dan membahayakan sistem jaringan *online* pada *provider* layanan sebagai bisnis mereka (*attacker*) yang tergantung pada ketersediaan *website* dari *provider* jaringan untuk membuat bisnis dari *provider* jaringan menjadi kritis dan produktivitas menurun drastis.

3.1.9 Resiko-resiko keamanan aplikasi

Menurut Williams dan Wichers (2013), penyerang berpotensi menggunakan beragam cara melalui aplikasi anda untuk membahayakan

bisnis atau organisasi anda. Setiap cara mewakili risiko, yang mungkin cukup serius untuk memperoleh perhatian seperti pada gambar 3.1.

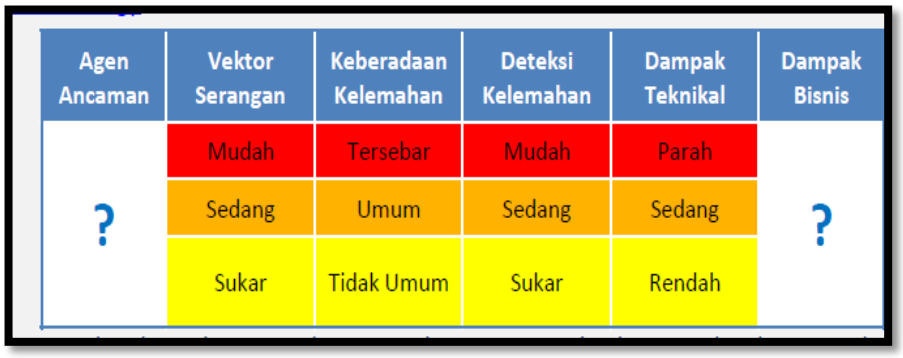


Sumber : https://www.*OWASP*.org/index.php/Top_10_2010-Main (diakses 9-10-2017)

Gambar 3.1 Alur serangan pada aplikasi.

Terkadang cara ini mudah ditemukaan dan dieksploitasi, namun kadang-kadang sulit. Demikian juga, kerusakan yang diakibatkan dapat berkisar dari tidak ada apa-apa hingga membuat anda keluar dari bisnis. Untuk menentukan resiko di organisasi anda, anda dapat mengevaluasi kemungkinan yang diasosiasikan untuk setiap agen ancaman, faktor serangan, kelemahan keamanan, dan mengkombinasikan dengan estimasi dampak teknis dan bisnis bagi organisasi anda. Semua faktor ini menentukan risiko keseluruhan.

Pembaruan OWASP (*The Open Web Application Security Project*) *Top Ten* ini berfokus pada identifikasi risiko yang paling serius bagi sebagian besar organisasi. Untuk setiap resiko, kami memberikan informasi umum mengenai kemungkinan dan dampak teknis dengan menggunakan skema penilaian sederhana sepert padai gambar 3.2 yang berdasarkan pada *OWASP* (*The Open Web Application Security Project*) *Risk Rating Methodology*.



Agen Ancaman	Vektor Serangan	Keberadaan Kelemahan	Deteksi Kelemahan	Dampak Teknikal	Dampak Bisnis
?	Mudah	Tersebar	Mudah	Parah	?
	Sedang	Umum	Sedang	Sedang	
	Sukar	Tidak Umum	Sukar	Rendah	

Sumber : https://www.*OWASP*.org/index.php/Top_10_2010-Main (diakses 9-10-2017)

Gambar 3.2 Skema penilaian dampak serangan.

Pengujian aplikasi untuk menemukan kerentanan keamanan tanpa mengetahui *inner* dari aplikasi itu sendiri disebut pengujian penetrasi, *tester* bertindak sebagai seorang penyerang dan berupaya untuk menemukan dan mengeksploitasi kerentanan. Kerentanan yang ditemukan dapat di kelompokkan menjadi tiga kategori yaitu *low*, *medium* dan *high*, tingkatan resiko kerentanan dihitung dari dua faktor yaitu faktor estimasi kemungkinan dan faktor estimasi dampak. Setiap faktor memiliki pilihan dan setiap pilihan mempunyai *rating* 0 - 9, *rating* ini akan digunakan untuk menghitung tingkat resiko setiap faktor. Dari *rating* akan didapatkan *level* kerentanan, berdasarkan *OWASP* tingkat resiko kerentanan dibagi tiga yaitu *low*, *medium*, *high* yang memiliki *rating* nilai berbeda-beda, seperti pada gambar 3.3 .

0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Gambar 3.3: Tingkat kemungkinan dan dampak

Faktor-faktor untuk estimasi kemungkinan berhubungan dengan bagaimana kerentanan ditemukan dan dieksploitasi oleh penyerang. Faktor ini dibagi menjadi dua kategori, yaitu *Threat Agent Factors* dan *Vulnerability Factors*. Kategori *Threat Agent Factors* memiliki empat faktor yaitu *Skill level, Motive, Opportunity, Size*, pada Tabel 3.3.

Tabel 3.3: Faktor kemungkinan *threat agent*

Faktor	Pilihan	Rating
Skill level	Security penetration skills	9
	Network and programming skills	6
	Advanced computer user	5
	Some technical skills	3
	No technical skills	1
Motive	Low or no reward	1
	Possible reward	4
	High reward	9
Opportunity	Full access or expensive resources required	0
	Special access or resources required	4
	Some access or resources required	7
	No access or resources required	9
Size	Developers	2
	System administrators	2
	Intranet users	4
	Partners	5
	Authenticated users	6
	Anonymous Internet users	9

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Kategori *Vulnerability Factors* memiliki empat faktor yaitu *Ease of discovery, Ease of exploit, Awareness, Intrusion detection* dan masing-masing faktor memiliki pilihan, pada Tabel 3.4.

Tabel 3.4: Faktor kemungkinan *vulnerability*

Faktor	Pilihan	Rating
Ease of discovery	Practically impossible	1
	Difficult	3
	Easy	7
	Automated tools available	9
Ease of exploit	Theoretical	1
	Difficult	3
	Easy	5
	Automated tools available	9
Awareness	Unknown	1
	Hidden	4
	Obvious	6
	Public knowledge	9
Intrusion detection	Active detection in application	1
	Logged and reviewed	3
	Logged without review	8
	Not logged	9

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Faktor untuk estimasi dampak dibagi menjadi dua kategori, yaitu *Technical Impact Factors* dan *Business Impact Factors*. Kategori *Technical Impact Factors* memiliki empat faktor yaitu *loss of confidentiality*, *loss of integrity*, *loss of availability* dan *loss of accountability*, pada Tabel 3.5.

Tabel 3.5: T*echnical impact factors*

Faktor	Jenis	Nilai
Loss of confidentiality	Minimal non-sensitive data disclosed .	2
	Minimal critical data disclosed.	6
	Extensive non-sensitive data disclosed.	6
	Extensive critical data disclosed.	7
	All data disclosed	9
Loss of integrity	Minimal slightly corrupt data.	1
	Minimal seriously corrupt data.	3
	Extensive slightly corrupt data.	5
	Extensive seriously corrupt data.	7
	All data totally corrupt.	9
Loss of availability	Minimal secondary services interrupted.	1
	Minimal primary services interrupted.	5
	Extensive secondary services interrupted.	5
	Extensive primary services interrupted.	7
	All services completely lost.	9
Loss of accountability	Fully traceable.	1
	Possibly traceable.	7
	Completely anonymous.	9

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Kategori *Business Impact Factors* memiliki empat faktor yaitu *financial damage*, *reputation damage*, *non-complicance* dan *privacy violation*, dan setiap faktor memiliki pilihan, Tabel 3.6.

Tabel 3.6: B*usiness impact factors*

Faktor	Jenis	Nilai
Financial damage	Less than the cost to fix the vulnerability	1
	Minor effect on annual profit	3
	Significant effect on annual profit	7
	Bankruptcy	9
Reputation damage	Minimal damage	1
	Loss of major accounts	4
	Loss of goodwill	5
	Brand damage	9
Non-compliance	Minor violation	2
	Clear violation	5
	High profile violation	7
Privacy violation	One individual	3
	Hundreds of people	5
	Thousands of people	7
	Millions of people	9

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Rating terakhir tingkat resiko kerentanan didapatkan dengan menggabungkan dua faktor yaitu faktor estimasi kemungkinan dan faktor estimasi dampak. Penentuan akhir tingkat resiko kerentanan menggunakan rumus seperti berikut.

Resiko = Kemungkinan * Dampak

Jika dari faktor estimasi dampak ditemukan untuk kategori dampak teknis dan kategori dampak bisnis, maka kategori dampak bisnis menjadi prioritas, namun jika kategori dampak bisnis tidak ditemukan maka menggunakan kategori dampak teknis. Dari perhitungan dua faktor

dengan menggunakan rumus diatas didapatkan hasil seperti pada gambar 3.4.

Dampak	TINGGI	Medium	Tinggi	Kritis
	MEDIUM	Rendah	Medium	Tinggi
	RENDAH	Catatan	Rendah	Medium
		RENDAH	MEDIUM	TINGGI
	Kemungkinan			

Sumber : *https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology*

Gambar 3.4. Keseluruhan tingkat resiko kerentanan

Namun demikian, hanya anda yang tahu mengenai lingkungan dan bisnis anda secara khusus. Untuk setiap aplikasi, mungkin tidak ada agen ancaman yang dapat melakukan serangan yang sesuai, atau dampak teknis tidak membuat perubahan. Karenanya, anda harus mengevaluasi setiap risiko, berfokus pada agen ancaman, kendali keamanan, dan dampak bisnis dalam perusahaan anda.

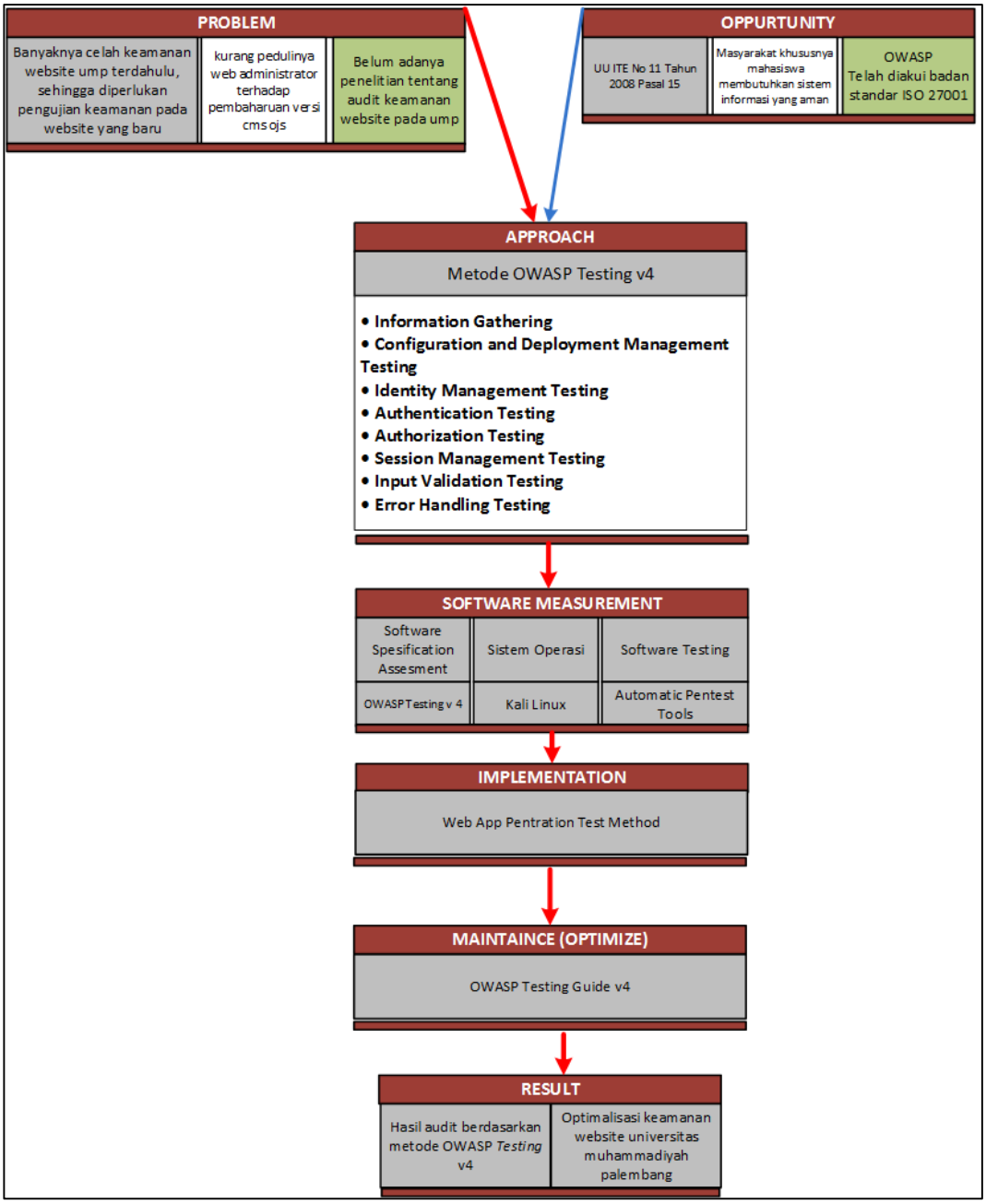
3.2 Hasil Penelitian Terdahulu

Penulis mengambil beberapa jurnal dan skripsi dari penelitian sebelumnya sebagai panduan untuk penyusunan skripsi, antara lain :

1. Purnama, dan Budi (2016) , Audit Keamanan *Web Server* Pada STMIK PalComTech Palembang, dengan hasil menemukan celah keamanan CSRF yang dienkripsi dengan menggunakan base64 pada krs *online* mahasiswa dan beberapa hasil *vulnerability* lainnya yang didapat dari hasil *scanning* menggunakan *tools* nikto.

2. Hidayatullah, dan Priadi (2016), Analisis Evaluasi Keamanan Pada *Website* STIK Siti Khadijah, dengan hasil mencoba penetrasi dengan teknik penyerangan seperti, SQL *Injection*, *Cross-Site Scripting* (XSS), *Denial of Service* (DOS) dan melakukan *upload webshell exploit* php (b374k) untuk *remote webserver* dari *browser*, dan berhasil menemukan *vulnerability* pada *upload file*.
3. Muhsin dan Fajaryanto (2015), Penerapan Pengujian Keamanan Web Server Menggunakan Metode OWASP versi 4 (Studi Kasus Web Server Ujian Online), memperoleh hasil penemuan celah keamanan berdasarkan panduan pada OWASP versi 4 dengan menggunakan *automated scanning tools* seperti webscrab, brutus, dirb, wfuzz, OSWAP ZAP(*zed attack proxy*) dan OWASP CSRF *Tester*.
4. Cobantoro, Adi Fajaryanto (2016). Penerapan Pengujian Keamanan Web Server Menggunakan Metode OWASP versi 4 (Studi Kasus E-Jurnal Server Kampus X Madiun), memperoleh hasil bahwa manajemen otentifikasi, otorisassi dan manajemen sesi belum diimplementasikan dengan baik.

3.3 Kerangka Penelitian



Sumber : Diolah sendiri.

Gambar 3.5 Kerangka Penelitian

Berdasarkan kerangka penelitian pada gambar 3.5 , peneliti menemukan tiga masalah yaitu

1. Banyaknya celah keamanan pada *website* universitas muhammadiyah palembang terdahulu sehingga diperlukan pengujian kemanan pada *website* yang baru
2. Kurang pedulinya web administrator dalam memperbaharui versi cms ojs
3. Belum adanya penelitian tentang audit kemanan *website* pada universitas muhammadiyah Palembang.

Peluang keberhasilnya penelitian ini didukung oleh :

1. Undang Undang Informasi Transaksi Elektronik No.11 Thn.2008 Pasal 15.
2. Masyarakat khususnya masasiswa membutuhkan sistem informasi yang aman
3. OWASP telah diakui oleh badan standar ISO 27001.

Pendekatan digunakan dalam penelitian ini adalah metode OWASP *Testing* v4 yang terdiri dari 8 subkategori pengujian yaitu :

1. *Information Gathering*
2. *Configuration and Deployment Management Testing*
3. *Identity Management Testing*
4. *Authentication Testing*
5. *Authorization Testing*
6. *Session Management Testing*
7. *Input Validation Testing*
8. *Error Handling Testing*

Software Pengukuran dan Pengujian yang digunakan adalah:

1. Buku OWASP *Testing v. 4* Tahun *2014* yang berguna untuk mengaudit dan membuat laporan.
2. Sistem Operasi Kali Linux adalah sistem operasi yang dikhususkan untuk melakukan pengujian *penetration testing.*.
3. *Automatic Penetration Tools* yang digunakan untuk melakukan pengujian keamanan *website.*

Pada tahapan Implementasi penelitian ini akan menggunakan metode *web application penetration testing.*

Pada tahapan optimalisasi, akan dilakukan berdasarkan hasil temuan audit baik itu dalam bentuk rekomendasi perbaikan maupun memperbaiki secara langsung atas izin yang bersangkautan.

Hasil penelitian akan didapatkan yaitu:

1. Hasil audit keamanan website menggunakan metode OWASP *Testing* v4.
2. Optimalisasi keamanan *website* berdasarkan hasil temuan audit keamanan *website.*