Màster en Seguretat de les Tecnologies de la Informació i les Comunicacions

Treball de Fi de Màster
Seguretat en Aplicacions Web

EINES DE DETECCIÓ DE VULNERABILITATS EN APLICACIONS WEB – CONTROL DE DADES D'ENTRADA

Josep Villar Azorín

Taula de Continguts

Introducció

- L'ús d'Internet ha augmentat de forma exponencial en la última dècada. Això ha provocat un increment del mal ús de les aplicacions disponibles des d'aquesta xarxa amb finalitats il·lícites com ara:
  - Obtenció de dades sensibles
  - Robatori (phishing)
  - Boicot a organitzacions
- La debilitat més comuna en la seguretat de les aplicacions web, és la falta de validació adequada de les entrades procedents del client o de l'entorn de l'aplicació. Aquesta debilitat condueix a quasi totes les principals vulnerabilitats en aplicacions.
- Les dades procedents de qualsevol entitat/client externs mai haurien de considerar-se confiables, ja que podria alterar aquestes dades: *All input is evil*.

Vulnerabilitats en l'entrada de dades 1

- Les dades d'entrada a les aplicacions, que normalment arriben en paràmetres o el cos de les peticions HTTP, són una font potencial de problemes de seguretat.
- Atès que les tecnologies relacionades amb Internet canvien constantment, les vulnerabilitats associades a aquestes també ho fan.
- Portar un control exhaustiu de les vulnerabilitats noves que poden aparèixer i donar informació, així com propostes de solució a aquestes és una tasca complexa, per lo que han aparegut una sèrie d'organismes a nivell internacions que se n'encarreguen.
- Un dels més reconeguts és l'OWASP (Open Web Application Security)

Vulnerabilitats en l'entrada de dades 2

- *OWASP* és una comunitat oberta que dedicada a permetre a les organitzacions realitzar el desenvolupament, adquisició i manteniment d'aplicacions confiables, de forma lliure i gratuïta.
- *OWASP* ofereix, de forma similar als projectes de codi obert, molts tipus de material amb un desenvolupament col·laboratiu i obert, com documentació, guies, llibreries i eines de desenvolupament i test.
- Un dels projectes d'*OWASP* és el *Testing Project*, que mitjançant la seva guia enumera aquells problemes de seguretat que s'han de tenir en compte per verificar que les aplicacions estan protegides contra les vulnerabilitats que representen.
- Hi ha un capítol dintre d'aquesta guia que es centra en les proves de validació de dades.

Vulnerabilitats en l'entrada de dades 3

- En la seva guia del *Testing Project, OWASP* organitza el capítol de les proves sobre les dades en els següents punts:
  - Injecció d'*scripts*
  - Injecció de codi
  - Injecció de comandes
  - *Buffer overflow*
  - Vulnerabilitat incubada
  - HTTP *splitting / smuggling*

Vulnerabilitats en l'entrada de dades 4

INJECCIÓ D'*SCRIPTS (XSS)*

- Parlem d'aquest tipus de prova quan s'intenten manipular les dades d'entrada que rep per paràmetre l'aplicació per tal que es generi una sortida maliciosa.
- Es dona quan no es valida l'entrada i es pot controlar la sortida generada.
- Aquesta vulnerabilitat genera diferents tipus d'atac:
  - *Reflected XSS (OWASP-DV-001)*
  - *XSS* emmagatzemat *(OWASP-DV-002)*
  - *XSS* basat en DOM *(OWASP-DV-003)*
  - *XSS* basat en XML *(OWASP-DV-004)*

Vulnerabilitats en l'entrada de dades 5

INJECCIÓ DE CODI

- Parlem d'aquest tipus de prova quan s'intenten manipular les dades d'entrada que rep per paràmetre l'aplicació per tal de fer arribar codi maliciós executable dintre del servidor.
- Aquesta vulnerabilitat genera diferents tipus d'atac:
  - *Injecció SQL (OWASP-DV-005)*
  - *Injecció LDAP (OWASP-DV-006)*
  - *Injecció ORM (OWASP-DV-007)*
  - *Injecció* XML *(OWASP-DV-008)*
  - *Injecció SSI (OWASP-DV-009)*
  - *Injecció XPath (OWASP-DV-010)*
  - *Injecció IMAP/SMTP (OWASP-DV-011)*
  - *Injecció de codi (OWASP-DV-012)*

Vulnerabilitats en l'entrada de dades 7

VULNERABILITATS INCUBADES *(OWASP-DV-015)*

- També conegudes com *atacs persistents*. Es tracta d'aconseguir emmagatzemar l'atac en el sistema (un *XSS* o una *Injecció de codi*) per tal de poder-lo explotar a posteriori.

HTTP SMUGGLING/SPLITTING *(OWASP-DV-016)*

- Atacs que aprofiten característiques pròpies del protocol HTTP, explotant debilitats de l'aplicació web o particularitats en què els agents (per exemple un navegador) interpreten els missatges HTTP.

Vulnerabilitats en l'entrada de dades 8

CONCLUSIONS

- Les dades d'entrada a una aplicació són una de les fonts de vulnerabilitats més importants.
- És important tractar adequadament les dades d'entrada. En el cas de paràmetre tant el seu contingut, tant en el propi nom del paràmetre.
- Entre les diferents tècniques que es poden aplicar pel tractament de les dades d'entrada trobem:
  - Eliminació de caràcters o seqüències de caràcters *sospitosos* (<, &, ', ...)
  - Ús de llistes negres
  - Ús de llistes blanques
  - Combinacions de les anteriors

Eines de detecció de vulnerabilitats 1

INTRODUCCIÓ

- Les eines de detecció de vulnerabilitats automatizades o *black box scanners*, ajuden a la comprovació de la seguretat d'una aplicació en funció d'una sèrie de proves que executen contra aquesta.
- Per norma general, no accedeixen al codi font de les aplicacions destí i han d'aplicar els potencials atacs que poden patir aquestes.
- Un dels factors importants amb el que aquestes eines demostren la seva fiabilitat és no només el poder descobrir les vulnerabilitats contemplades si no també la seva habilitat per diferenciar falsos positius de les vulnerabilitats reals.
- Aquestes eines no fan les aplicacions més segures, si no ajuden a ser-ho i a aplicar polítiques de seguretat.

Eines de detecció de vulnerabilitats 2

ESPECIFICACIONS FUNCIONALS

- En base al document *Software Assurance Tools: Web Scanner Functional Specification 1.0*, el projecte *SAMATE* del *NIST* una eina d'aquest tipus ha de contemplar els següents requeriments funcionals:
  - Contemplar un conjunt mínim de vulnerabilitats
  - Informar d'un atac que demostra cada vulnerabilitat
  - Especificar com s'ha produït un atac
  - De forma opcional:
    - No identificar vulnerabilitats *obsoletes*
    - Fer propostes de com evitar un atac
    - Oferir els informes de resultat en format XML
    - Assignar una criticitat a cada vulnerabilitat

Eines de detecció de vulnerabilitats 2

CRITERIS D'AVALUACIÓ

- Amb la finalitat de poder avaluar les eines, el *WASC* ha publicat una guia d'avaluació que comprèn els següents punts:
  - Protocols suportats tant a nivell HTTP com *Proxy*
  - Mètodes d'autenticació suportats
  - Gestió de les sessions
  - *Crawling*
  - Anàlisi *(Parsing)*
  - Capacitat de *Testing*
  - Interfícies de comandes, control i configuració
  - Producció d'informes. Inclou personalització i diferents formats de sortida
  - Assessorament sobre les vulnerabilitats trobades

Eines de detecció de vulnerabilitats 3

EFECTIVITAT DE LES EINES

- Atesa la gran quantitat d'eines existents, tant comercials com de codi obert i de *software-as-a-service*, no es pot parlar de forma general sobre l'efectivitat d'aquestes eines. A més, dependrà molt de la tipologia de l'aplicació i les necessitats concretes de cada cas.
- Per tant, altres aspectes que han d'ajudar a determinar l'efectivitat d'una eina a part de les exposades en el punt anterior, són:
  - Fer èmfasi especial en la fase de *crawling* i *parsing*
  - El temps d'execució i l'ample de banda consumit
  - El ventall final de les vulnerabilitats detectades
  - La gestió de falsos positius

Eines de detecció de vulnerabilitats 4

CONCLUSIONS

- Tot i que, a priori, no podrien semblar molt efectives en funció del percentatge de vulnerabilitats trobades, son de gran utilitat per corregir problemes de seguretat en les aplicacions web, així com establir un control de qualitat. Resoldre problemes per una vulnerabilitat determinada pot implicar resoldre problemes amb d'altres vulnerabilitats no detectades en millorar la qualitat de les dades d'entrada.
- Una àrea de millora general en aquestes eines és la capacitat de detecció de vulnerabilitats de tipus *diferit* (*XSS*, *SQLI*).
- Aquestes eines s'han d'utilitzar de forma periòdica. Tant les vulnerabilitats com aquestes eines evolucionen amb el temps.
- Tot i que hi ha guies d'avaluació d'aquestes eines, s'ha de triar la que més s'ajusti a les necessitats especials de l'aplicació a validar.

SKIPFISH 1

- *Skipfish* és un projecte de Google Code amb llicència Apache 2.0 que consisteix en una eina de detecció de vulnerabilitats en aplicacions Web .
- Prepara un mapa del lloc web a verificar a partir d'una cerca recursiva i exploracions basades en diccionari. A partir d'aquest mapa aplica els tests de seguretat per trobar les vulnerabilitats associades.
- Està escrit en llenguatge C pur, fet que proporciona més velocitat i eficiència que en d'altres eines escrites en llenguatges interpretats o semi-interpretats.
- Incorpora una gestió optimitzada del protocol HTTP que suposa baix impacte en CPU, sent capaç d'executar nombres molt elevats de peticions per segons, depenent de l'arquitectura de la xarxa.

SKIPFISH 2

- Capacitats d'atacs de força bruta basada en diccionaris. Es contemplen 3 modalitats: sense força bruta, força bruta lleugera i força bruta *normal*.
- Comprovacions de seguretat d'alta qualitat (segons els autors).
- Interfície d'usuari basada en línia de comandes.
- Producció d'informes finals molt senzills basats en HTML/Javascript. Categoritza les vulnerabilitats en 5 nivells de criticitat:
  - Alta
  - Mitja
  - Baixa
  - Avisos
  - Informació addicional

SKIPFISH 3

- *Skipfish* és un projecte viu en evolució, i tot i la seva eficàcia encara hi ha una sèrie d'aspectes a treballar i millorar:
  - Millora en la detecció de *buffer overflows*
  - Extendre el *crawler* a més tipus de continguts que no siguin HTML.
  - Ampliar el ventall de comprovacions de força bruta, especialment pel que fa a passwords.
  - Contemplar altres mètodes d'autenticació addicionals a BASIC i basada en formulari, com NTLM i DIGEST.
  - Suport a *proxies* HTTP i SOCKS
  - Possibilitat de reprendre una execució aturada prèviament.
  - Sistema d'instal·lació més automatitzat
  - Millora en la interfície d'usuari i dels informes obtinguts

SKIPFISH 4

- Després de fer un test sobre una aplicació amb vulnerabilitats de forma deliberada (WebGoat del projecte OWASP), es pot constatar que l'eficàcia de l'eina és força alta, detectant un bon número de les vulnerabilitats en un interval de temps raonablement curt.
- Tot i això, com s'ha comentat anteriorment, hi ha àrees de millora a solventar.
- La millora d'algunes d'aquestes àrees en especial és el que es proposa en el present treball. Aquestes estan relacionades amb els següents aspectes:
  - Interfície d'entrada
  - Resultats obtinguts
- A tal efecte, es presenta a continuació una interfície Web que permet una interacció més còmoda a l'usuari i una producció d'informes més extensa.

INTERFÍCIE WEB 1

- Proposta d'aplicació per millorar els aspectes d'interfície d'usuari i dels informes de resultat.
- Es tracta d'una aplicació Web desenvolupada amb tecnologia JEE i Spring Framework.
- Aquesta aplicació inicialment permetrà:
  - Mantenir definicions de nous tests que quedaran emmagatzemats en una base de dades relacional.
  - Fer execucions dels tests a partir de les definicions
  - Obtenir informes més complets dels que proporciona originalment *Skipfish* on es proporciona informació addicional sobre les vulnerabilitats i propostes de solució a les mateixes.
  - Generar informes de comparació entre dues execucions d'una mateixa definició per poder comprovar els progressos en la resolució de vulnerabilitats.

INTERFÍCIE WEB 2

FUNCIONALITATS I

- Alta de definició. Permet donar d'alta una nova definició d'un test. Es presenta a l'usuari un formulari amb gairebé totes les opcions disponibles a la línia de comandes d'*Skipfish*. Aquest formulari organitza els camps en funció de les diferents agrupacions de paràmetres que existeixen i realitza les validacions necessàries per a cada paràmetre.
- Llistat de definicions. Es presenta una llista amb les definicions disponibles. Cada definició té associada una sèrie d'accions sobre ella:
  - Modificar
  - Eliminar
  - Executar test
  - Consultar tests realitzats.

INTERFÍCIE WEB 3

FUNCIONALITATS II

- Modificació d'una definició. Es mostra el mateix formulari de l'alta amb les dades d'una definició prèviament informades. Es fan les mateixes comprovacions que en una alta.
- Eliminació d'una definició prèvia confirmació per part de l'usuari. Esborra la definició triada i torna a presentar la llista de definicions.
- Execució d'un test. Es presenta un formulari on s'introdueix una descripció pel test a executar i una llista opcionals d'adreces de mail per notificar la finalització del test. A continuació és quan es crida *Skipfish* construint la línia de comandes a partir de les dades que la definició té emmagatzemades.
- Consultar tests. Es presenta una nova plana amb la llista de tests disponibles. Per a cada execució hi ha disponibles les opcions de consultar, eliminar i, en triar-ne dues, comparar-les.

INTERFÍCIE WEB 4

FUNCIONALITATS III

- Eliminació d'un test prèvia confirmació per part de l'usuari. Esborra l'execució triada i torna a presentar la llista d'execucions de la definició en curs.
- Consulta d'un test. Mostra l'informe corresponent al test generat per *Skipfish* però completat amb informació addicional sobre cada vulnerabilitat i amb propostes de resolució per a cadascuna d'elles.
- Comparar tests. Cada fila de la llista de tests incorpora un *checkbox*. En triar dues execucions, es dóna la possibilitat de comparar-les. En aquest cas es presenta un informe on es detallen les vulnerabilitats ja solucionades i les noves que hagin pogut sorgir entre les dues execucions triades.

INTERFÍCIE WEB 5

ÀREES DE MILLORA

- Finalització de detalls d'implementació.
- Securització de l'aplicació.
- Millora en la interfície gràfica i la usabilitat.
- Generació d'informes més complerts i oferir més formats de sortida disponibles.
- Sistema de planificació de les execucions
- Millores en la gestió de les definicions i execucions. Poder aplicar patrons de cerca, paginacions, etc.
- Eines per a la gestió de la informació complementària. Poder mantenir aquesta informació amb un editor HTML des de dins la mateixa aplicació.
- Comprovar que la pròpia aplicació és segura i no vulnerable.