ME/PSO

DORACAK

Doracak për sigurinë kibernetike për programuesit e TI-së

Programim i sigurt dhe mbrojtje gjatë zhvillimit të softuerit

Material shoqërues për trajnimin:
„Cybersecurity Awareness for IT Developers"

Dokument nr. MEPSO-PRI-DEV-006 · Klasifikimi: Përdorim i brendshëm
Sektori i sigurisë informatike dhe kibernetike · MEPSO SHA Shkup

Përmbajtja

1. Hyrje

Softueri që e krijojmë është themeli i infrastrukturës digjitale. Çdo rresht kodi mund të jetë ose mbrojtje ose dobësi. Siguria duhet të jetë pjesë e zhvillimit që nga fillimi, jo shtesë në fund.

Ky doracak është i dedikuar për programuesit dhe zhvilluesit. Do të mbulojmë parime të programimit të sigurt, dobësitë më të shpeshta dhe praktika për krijimin e softuerit të besueshëm. Qëllimi është që siguria të bëhet pjesë e natyrshme e procesit tuaj të zhvillimit.

Si organizatë që menaxhon infrastrukturë kritike, cilësia dhe siguria e softuerit që e krijojmë dhe e përdorim kanë pasoja të drejtpërdrejta. Programimi i sigurt nuk është opsion — ai është përgjegjësi profesionale.

> **Parimi kyç.** Siguria nuk mund të „shtohet" në fund. Ajo duhet të ndërtohet në dizajn dhe në implementim që nga fillimi. Rregullimi i një dobësie sigurie në produksion është shumëfish më i shtrenjtë se parandalimi i saj gjatë zhvillimit.

2. Siguria si pjesë e zhvillimit

Programimi i sigurt do të thotë të mendosh për sigurinë në çdo fazë të zhvillimit.

2.1 Siguria sipas dizajnit

- ✓ Mendoni për kërcënimet që në fazën e dizajnit
- ✓ Zbatoni parimin e privilegjit më të vogël
- ✓ Supozoni se të dhënat hyrëse janë të pabesueshme
- ✓ Planifikoni për dështim — si reagon sistemi ndaj gabimeve
- ✓ Mbrojtje në thellësi — disa shtresa mbrojtjeje

2.2 Mënyra e të menduarit

Programuesi i mirë mendon jo vetëm „si do të punojë kjo", por edhe „si mund të keqpërdoret kjo". Kjo mënyrë e të menduarit për sigurinë është kyçe për krijimin e softuerit të besueshëm.

> **E rëndësishme.** Çdo hyrje nga përdoruesi, çdo lidhje e jashtme, çdo varësi është pikë potenciale sulmi. Trajtojini me kujdesin e duhur.

3. Dobësitë më të shpeshta të sigurisë

Kuptimi i dobësive të shpeshta është hapi i parë drejt parandalimit të tyre.

Dobësia	Rreziku
Injektimi	Ekzekutim i komandave keqdashëse përmes të dhënave hyrëse
Autentifikim i dobët	Qasje e paautorizuar në sistem
Ekspozimi i të dhënave	Zbulim i informacioneve të ndjeshme
Konfigurim i gabuar	Dobësi sigurie për shkak të cilësimeve të këqija
Varësi të cenueshme	Dobësi të trashëguara nga komponentët e jashtëm
Validim i pamjaftueshëm	Pranim i të dhënave hyrëse keqdashëse

4. Validimi i të dhënave hyrëse

Kurrë mos u besoni të dhënave hyrëse. Çdo hyrje — nga përdoruesi, skedari, rrjeti ose një sistem tjetër — duhet të validohet.

4.1 Parimet e validimit

- ✓ Validoni në anën e serverit, jo vetëm në atë të klientit
- ✓ Zbatoni qasjen „allow-list" — lejoni vetëm atë që dihet se është e mirë
- ✓ Kontrolloni llojin, gjatësinë, formatin dhe diapazonin
- ✓ Refuzoni — mos u përpiqni t'i „rregulloni" — të dhënat e dyshimta
- ✓ Validoni sa më afër pikës së hyrjes

Rregulli i artë. Çdo e dhënë hyrëse është potencialisht keqdashëse derisa të vërtetohet e kundërta. Validimi i rreptë është linja e parë e mbrojtjes kundër shumë llojeve të sulmeve.

5. Menaxhimi i sigurt i fjalëkalimeve dhe sekreteve

Fjalëkalimet, çelësat dhe sekretet e tjera kërkojnë mbrojtje të veçantë në kod.

5.1 Rregullat

- ✓ Kurrë mos i vendosni sekretet drejtpërdrejt në kod
- ✓ Mos ruani fjalëkalime në tekst të hapur
- ✓ Përdorni metoda të sigurta për ruajtjen e fjalëkalimeve
- ✓ Mbajini sekretet në sisteme të mbrojtura konfigurimi
- ✓ Mos i përfshini sekretet në sistemin e versioneve

Gabim i shpeshtë. Vendosja e fjalëkalimeve, çelësave API ose sekreteve të tjera drejtpërdrejt në kodin burimor. Nëse kodi ndahet ose rrjedh, sekretet ekspozohen. Gjithmonë përdorni metoda të sigurta për menaxhimin e sekreteve.

6. Autentifikimi dhe autorizimi

Menaxhimi i drejtë i identitetit dhe i qasjes është themeli i softuerit të sigurt.

6.1 Autentifikimi

- ✓ Përdorni mekanizma të provuar, standardë për autentifikim
- ✓ Mbështetni autentifikimin me dy faktorë ku është e mundur
- ✓ Mbrohuni nga sulmet me hamendësim të fjalëkalimeve
- ✓ Menaxhojini sesionet në mënyrë të sigurt

6.2 Autorizimi

- ✓ Kontrollojini lejet për çdo veprim
- ✓ Zbatoni privilegjin më të vogël
- ✓ Mos u mbështetni vetëm në fshehjen e funksioneve në ndërfaqe
- ✓ Validojeni autorizimin në anën e serverit

7. Mbrojtja nga injektimet

Injektimet janë ndër dobësitë më të rrezikshme — sulmuesi fut kod keqdashës që sistemi e ekzekuton.

7.1 Llojet e injektimeve

- **SQL injektimi** — manipulim i bazave të të dhënave;
- **Command injektimi** — ekzekutim i komandave të sistemit;
- **Cross-site scripting (XSS)** — injektim i skripteve në ueb-faqe.

7.2 Mbrojtja

- ✓ Përdorni pyetje të parametrizuara për bazat e të dhënave
- ✓ Validoni dhe „escape"-oni të dhënat dalëse
- ✓ Mos përpiloni komanda me të dhëna hyrëse të pabesueshme
- ✓ Zbatoni kodim të përshtatshëm të daljes

> **Kyçe.** Kurrë mos përpiloni pyetje ose komanda me bashkim të drejtpërdrejtë të të dhënave hyrëse. Gjithmonë përdorni metoda të sigurta, të parametrizuara.

8. Komunikimi i sigurt

- ✓ Përdorni komunikim të enkriptuar për të dhëna të ndjeshme
- ✓ Validojini certifikatat si duhet
- ✓ Mos transmetoni të dhëna të ndjeshme në tekst të hapur
- ✓ Zbatoni protokolle të sigurta, shmangni të vjetruarat

9. Menaxhimi i varësive

Softueri modern përdor shumë biblioteka të jashtme. Çdo varësi sjell dobësi potenciale.

- ✓ Përdorni vetëm biblioteka të besueshme, të mirëmbajtura
- ✓ Përditësojini varësitë rregullisht
- ✓ Ndiqni për dobësi të publikuara në bibliotekat e përdorura
- ✓ Minimizojeni numrin e varësive
- ✓ Kontrollojini varësitë para përfshirjes

Rreziku nga varësitë. Një dobësi në bibliotekë të jashtme bëhet dobësi në softuerin tuaj. Mbajtja e varësive të përditësuara dhe ndjekja e njoftimeve të sigurisë është kyçe.

10. Menaxhimi i gabimeve dhe logimi

10.1 Menaxhim i sigurt i gabimeve

- ✓ Mos zbuloni detaje të ndjeshme në mesazhet e gabimeve
- ✓ Logojini gabimet për analizë, por në mënyrë të sigurt
- ✓ Mos ua ekspozoni strukturën e brendshme përdoruesve
- ✓ Trajtojini gabimet në mënyrë të kontrolluar

10.2 Logim i sigurt

- ✓ Mos logoni të dhëna të ndjeshme (fjalëkalime, të dhëna personale)
- ✓ Logoni ngjarje relevante për sigurinë
- ✓ Mbrojini logjet nga qasja e paautorizuar

Kujdes. Mesazhet e detajuara të gabimeve, të dobishme për debagim, mund t'u zbulojnë informacione të ndjeshme sulmuesve. Në produksion shfaqni mesazhe të përgjithshme, kurse detajet ruajini në logje të mbrojtura.

11. Ruajtja e sigurt e kodit

- ✓ Mos përfshini sekrete në sistemin e versioneve
- ✓ Kontrollojeni qasjen në kodin burimor
- ✓ Shqyrtojeni kodin për dobësi sigurie
- ✓ Përdorni praktika të sigurta bashkëpunimi
- ✓ Përfshini kontrolle sigurie në procesin e zhvillimit

12. Testimi i sigurisë

Testimi i sigurisë i zbulon dobësitë para sulmuesve.

- ✓ Testojini të dhënat hyrëse me vlera të papritura
- ✓ Shqyrtojeni kodin për probleme sigurie
- ✓ Përdorni mjete për analizë automatike
- ✓ Testojini autentifikimin dhe autorizimin

13. Skenarë praktikë

Skenari 1: Hyrje e përdoruesit në bazë të të dhënave

Zhvilloni një funksion që pranon kërkimin e përdoruesit dhe e përdor për pyetje në bazën e të dhënave.

Qasja e drejtë: Përdorni pyetje të parametrizuara, kurrë nuk e bashkoni drejtpërdrejt hyrjen e përdoruesit në SQL. E validoni hyrjen para përdorimit. Kjo e parandalon SQL injektimin.

Skenari 2: Ruajtja e çelësit API

Aplikacioni duhet të lidhet me një shërbim të jashtëm duke përdorur çelës API.

Qasja e drejtë: Nuk e vendosni çelësin në kod. E ruani në sistem të mbrojtur konfigurimi ose në variabël të mjedisit, jashtë sistemit të versioneve. Kështu çelësi mbetet i mbrojtur edhe nëse kodi ndahet.

Skenari 3: Shfaqja e gabimit

Aplikacioni has në gabim në bazën e të dhënave gjatë përpunimit të një kërkese.

Qasja e drejtë: Përdoruesit i shfaqni mesazh të përgjithshëm. Detajet e gabimit (që mund të zbulojnë strukturë të brendshme) i logoni në mënyrë të sigurt për analizë. Nuk ekspozoni detaje teknike që sulmuesi mund t'i shfrytëzojë.

14. Pyetjet më të shpeshta

Pse validimi në anën e klientit nuk mjafton?

Sepse sulmuesit mund ta anashkalojnë. Validimi i klientit është për përvojën e përdoruesit; validimi i sigurisë duhet të jetë në anën e serverit, ku sulmuesi nuk mund ta manipulojë.

A është e sigurt t'i ruaj fjalëkalimet nëse janë të enkriptuara?

Fjalëkalimet duhet të ruhen me metoda të sigurta, njëkahëshe, të dizajnuara posaçërisht për fjalëkalime, jo me enkriptim të zakonshëm. Kurrë në tekst të hapur.

Sa shpesh duhet t'i përditësoj varësitë?

Rregullisht, dhe menjëherë kur publikohet dobësi sigurie në një bibliotekë të përdorur. Ndiqni njoftimet e sigurisë për varësitë tuaja.

Cili është parimi më i rëndësishëm i sigurisë në programim?

Kurrë mos u besoni të dhënave hyrëse dhe ndërtojeni sigurinë që nga fillimi. Këto dy parime parandalojnë pjesë të madhe të dobësive të shpeshta.

A e ngadalëson zhvillimin testimi i sigurisë?

Afatshkurt kërkon përpjekje, por afatgjatë kursen shumë. Rregullimi i një dobësie në produksion është shumëfish më i shtrenjtë se parandalimi gjatë zhvillimit.

Fjalorth i termave

Termi	Kuptimi
Injektimi	Futje e kodit keqdashës që sistemi e ekzekuton
SQL injektimi	Injektim i drejtuar ndaj bazave të të dhënave
XSS	Cross-site scripting — injektim i skripteve në ueb-faqe
Validimi	Kontroll që të dhënat hyrëse janë të sakta dhe të sigurta
Autentifikimi	Konfirmimi i identitetit
Autorizimi	Përcaktimi i veprimeve të lejuara
Pyetja e parametrizuar	Mënyrë e sigurt e pyetjes në bazë pa injektim
Varësia	Bibliotekë e jashtme që e përdor softueri
Privilegji më i vogël	Dhënia vetëm e të drejtave të domosdoshme

Lista e kontrollit të sigurisë për zhvilluesit

Gjatë dizajnit

- ✓ Mendova për kërcënimet potenciale
- ✓ Zbatova parimin e privilegjit më të vogël
- ✓ Planifikova për trajtim të sigurt të gabimeve

Gjatë implementimit

- ✓ I validoj të gjitha të dhënat hyrëse në anën e serverit
- ✓ Përdor pyetje të parametrizuara për bazat e të dhënave
- ✓ Nuk vendos sekrete në kod
- ✓ Zbatoj menaxhim të sigurt të fjalëkalimeve

✓ Nuk ekspozoj detaje të ndjeshme në gabime

Para produksionit

✓ E shqyrtova kodin për dobësi sigurie

✓ I përditësova varësitë

✓ Testova me të dhëna hyrëse të papritura

✓ Kontrollova autentifikimin dhe autorizimin

✓ Nuk përfshiva sekrete në sistemin e versioneve

> **Zakoni mbi të gjitha.** Kur kontrollet e sigurisë bëhen pjesë e natyrshme e rrjedhës suaj të punës, ato nuk ju ngadalësojnë — ato bëhen automatike. Programuesit më të mirë mendojnë për sigurinë pa pasur nevojë t'u kujtohet.

15. Përfundim

Programimi i sigurt është përgjegjësi profesionale. Çdo rresht kodi që e shkruani mund ta forcojë ose ta dobësojë softuerin. Duke i ndërtuar parimet e sigurisë në procesin tuaj të zhvillimit, krijoni softuer të besueshëm që e mbron organizatën dhe përdoruesit.

> **Mesazhi kyç.** Siguria nuk është punë e dikujt tjetër — ajo është përgjegjësi e çdo programuesi. Mendoni për sigurinë nga rreshti i parë i kodit deri në produksion, dhe softueri juaj do të jetë pjesë e zgjidhjes, jo e problemit.

MEPSO SHA Shkup — Operatori i sistemit të transmetimit të energjisë elektrike të Maqedonisë së Veriut
Dokument nr. MEPSO-PRI-DEV-006 · Klasifikimi: Përdorim i brendshëm