NÚKIB

Národní úřad
pro kybernetickou
a informační
bezpečnost

BEZPEČNOSTNÍ DOPORUČENÍ PRO VÝVOJ OTEVŘENÉHO SOFTWARU VE VEŘEJNÉ SPRÁVĚ

Národní úřad
pro kybernetickou
a informační bezpečnost

NÚKIB

Č.J. NEPŘIDĚLENO • BRNO • 5. DUBNA 2022

VERZE DOKUMENTU: 1.0

Následující doporučení není pro veřejnou správu právně závazné a nenahrazuje požadavky zákona o kybernetické bezpečnosti.

Obsah

1 Úvod 4

2 Organizační opatření 5

O.1 Před začátkem vývoje je zvážen výběr použitého jazyka a frameworku z hlediska bezpečnosti 5

O.2 Zdrojový kód je zveřejněn co nejdříve 5

O.3 Součástí repozitáře je soubor SECURITY 5

O.4 Je určena osoba zodpovědná za nahlášené zranitelnosti 5

O.5 Nahlášené zranitelnosti jsou opraveny do 90 dnů 6

O.6 Všechny opravené zranitelnosti jsou uvedeny v souboru se změnami 6

O.7 Účty vývojářů při autentizaci používají vícefaktorovou autentizaci 6

O.8 Účty vývojářů jsou svázány s pracovní e-mailovou adresou 7

O.9 Dokumentace je součástí repozitáře 7

O.10 Pro knihovny: Zranitelné verze knihoven jsou označeny 7

O.11 Neudržované aplikace a knihovny jsou označeny 7

O.12 Pro aplikace: Výchozí konfigurace je restriktivní 7

3 Správa a tvorba kódu 9

S.1 Zdrojový kód je verzován (VCS) a zveřejněn v otevřeném repozitáři 9

S.2 Pro vývoj se používají oddělené větve, které se následně slučují do hlavní vývojové větve 9

S.3 Je prováděna kontrola změn kódu 9

S.4 Repozitář neobsahuje binární spustitelné soubory nebo kompilované kódy 9

S.5 U dynamicky typovaných jazyků je využíváno striktní typování 9

4 Použité knihovny 10

D.1 Aplikace a knihovny využívají udržované závislosti 10

D.2 Preferovány jsou knihovny s bezpečným API 10

D.3 Preferovány jsou knihovny, jejichž autoři „dbají na bezpečnost” 11

D.4 Pro aplikace: Aplikace při definování závislostí používají „lock file” 11

D.5 Závislosti jsou stahovány z důvěryhodných úložišť 11

5 Kontinuální integrace 12

CI.1: Pro aplikace: Jsou kontrolovány verze použitých závislostí 12

CI.2: Nalezené zranitelnosti jsou testovány v rámci CI 12

CI.3: Pro aplikace: Sestavení aplikace je plně automatizované 12

CI.4: Je definován a vynucován standard pro zdrojový kód 12

CI.5: Jsou prováděny jednotkové nebo integrační testy v oblastech s vlivem na bezpečnost . 12
CI.6: Jsou prováděny automatizované bezpečnostní testy . 13
CI.7: Je prováděna kontrola tajných identifikátorů ve zdrojovém kódu . 13
CI.8: Pro aplikace: Vydané verze jsou podepsány . 14
6 Kryptografické prostředky . 15
C.1 Jsou využívány ověřené kryptografické knihovny . 15
C.2 Jsou využívány odolné kryptografické prostředky . 15
C.3 Při použití protokolu TLS je podporována alespoň verze 1.2 . 15
C.4 Je prováděno ověřování použitého certifikátu . 15
C.5 Aplikace používá systémové úložiště důvěryhodných certifikátů . 16
C.6 Uložená hesla jsou odolná proti offline útokům . 16
C.7 Porovnání hašů uložených hesel je odolné na časovou analýzu . 16
C.8 Ke generování náhodných tokenů jsou použity kryptograficky bezpečné pseudonáhodné generátory . 17
C.9 Tajné identifikátory jsou odstraňovány z paměti . 17
7 Záznamy o událostech . 18
L.1 Pro knihovny: Knihovny využívají standardní rozhraní pro logování . 18
L.2 Pro aplikace: Aplikace umožňuje logování důležitých akcí . 18
L.3 Pro aplikace: Aplikace podporuje napojení na centrální log management . 18
L.4 Nejsou zaznamenávány tajné identifikátory . 19
L.5 Výjimky jsou řízeny . 19
8 Relační databáze . 20
D.1 Jsou využívány primární klíče . 20
D.2 Jsou používány cizí klíče při vazbě na číselníkové hodnoty anebo na jiné entity . 20
D.3 V rámci databáze jsou kontrolovány hodnoty parametrů („constraints”) . 20
D.4 Pokud databázový sloupec předpokládá unikátní hodnotu, je využit unikátní index . 20
D.5 Při vkládání dat jsou využívány transakce . 20
D.6 Databázové schéma je komentováno . 20
9 Podmínky využití informací . 21
10 Příloha: Příklady souborů SECURITY a security.txt . 22
Příklad souboru SECURITY: . 22
Příklad souboru security.txt: . 22

1 Úvod

Software s otevřeným kódem přináší určité bezpečnostní výhody. Hůře se v něm skrývají záměrná „zadní vrátka“. Kód může procházet více bezpečnostních analytiků, a tak odhalit i méně zjevné zranitelnosti. Tvůrci otevřeného kódu si obvykle dávají více záležet na jeho kvalitě z důvodu možného reputačního rizika.

Na druhou stranu ale je kód otevřen i případným útočníkům, a to zjednodušuje jejich práci. Například u serverové aplikace si můžou ještě před zneužitím přesně ověřit, zda je zranitelnost zneužitelná a jak ji zneužít s co nejmenší pozorností monitorovacích nástrojů.

Proto, pokud se organizace rozhodne zveřejnit zdrojový kód, měla by zvážit možné přínosy a taktéž rizika z toho plynoucí. Pro snížení možných rizik slouží taktéž toto bezpečnostní doporučení. Jeho cílem je snížit množství potenciálních zranitelností („secure by design“) a v případě, že se v kódu nějaká zranitelnost objeví, tak aby byla co nejdříve opravena.

Tento dokument je určen vývojářům a osobám zabývajícím se kybernetickou bezpečnostní ve veřejné správě nebo společnostem dodávajícím veřejné správě software. Všechna doporučení jsou nezávazná a je na organizaci, které z nich a v jaké míře bude u svých projektů využívat.

2 Organizační opatření

O.1 Před začátkem vývoje je zvážen výběr použitého jazyka a frameworku z hlediska bezpečnosti

Pokud je projekt v přípravné fázi a teprve probíhá volba použitého jazyka a frameworku, jedním ze zvažovaných bodů by měla být taktéž bezpečnost použitých technologií. Z tohoto pohledu je vhodné preferovat moderní jazyky, které minimalizují problémy se zranitelnostmi na úrovni paměti (bez manuálního uvolňování paměti) a souběhů.

Při výběru frameworku je vhodné preferovat ty, které splňují pravidla pro bezpečná API (viz D.2) a jejichž autoři garantují opravy bezpečnostních chyb u použité verze po akceptovatelnou dobu. V případě dlouhodobého projektu s dlouhou plánovanou životností je vhodné využít takové verze, které nabízí dlouhodobou podporu.

Budoucí změna jazyka, frameworku nebo jen aktualizace na novější hlavní verzi frameworku bývá často velmi nákladná a vyžaduje přepsání velké části aplikace.

O.2 Zdrojový kód je zveřejněn co nejdříve

Zdrojový kód vyvíjené aplikace je zveřejněn v repozitáři co nejdříve, tak aby vývojáři s kódem od začátku pracovali jako s veřejným a taktéž aby bezpečnostní chyby mohly být objeveny co nejdříve.

Pokud nelze zdrojový kód zveřejnit už od začátku vývoje, vývojáři při vývoji musí pracovat stejně, jako by již byl veřejný. Tedy aby se v historii repozitáře neobjevily části, které měly zůstat neveřejné (přístupové klíče apod.).

O.3 Součástí repozitáře je soubor SECURITY

Soubor SECURITY obsažený v repozitáři je standardní způsob, jak informovat uživatele a bezpečnostní analytiky, jakým mají být hlášeny bezpečnostní chyby. Pro hlášení musí být využit neveřejný kanál (doporučujeme využít buď neveřejné issues v rámci nástroje pro sdílení kódu nebo e-mailový kontakt se zveřejněným veřejným PGP klíčem). V případě e-mailu by se mělo jednat o ne-jmenný kontakt (například `security@organizace.cz`) pro zajištění funkčnosti i v případě změny pracovníků. Soubor taktéž může obsahovat další informace, jako např. jaké verze jsou podporované a plánovanou dobu podpory. Tyto informace jsou uvedeny v anglickém jazyce, volitelně doplněné českou alternativou. Cílem opatření je zvýšit pravděpodobnost, že zranitelnost bude nahlášena tvůrci kódu, než aby informace o ní byly zveřejněny nebo zneužity.

V případě webové aplikace výchozí instalace aplikace obsahuje taktéž soubor `.well-known/security.txt` obsahující informace dle standardu uvedeného na securitytxt.org, který si každý správce může nahradit vlastním obsahem.

O.4 Je určena osoba zodpovědná za nahlášené zranitelnosti

V rámci organizace spravující zdrojový kód v repozitáři je určena odpovědná osoba, která bude reagovat na nahlášené zranitelnosti. V případě, že tato osoba nebude dostupná po delší dobu, je

určen její zástup. Na tyto osoby by měla být směřována e-mailová adresa uvedená v souboru SECURITY a `security.txt`.

O.5 Nahlášené zranitelnosti jsou opraveny do 90 dnů

Všechny nalezené a nahlášené zranitelnosti musí být opraveny do 90 dnů, včetně vydání nové verze opravující tuto chybu. Lhůta může být prodloužena v případě zranitelností, které vyžadují např. změnu architektury aplikace. U nahlášené zranitelnosti externím subjektem tak ale může být učiněno pouze po domluvě s nahlašovatelem zranitelnosti – bezpečnostní výzkumníci obvykle informaci o zranitelnosti zveřejní, pokud není opravena do předem domluvené doby.

V případě kritické zranitelnosti (např. RCE zneužitelné bez potřeby autentizace, CVSS vyšší než 9.0) by měla být oprava do kódu začleněna v rámci hodin před vydáním nové verze. Zveřejnění opravy dává útočníkovi informaci, ve které části aplikace je zranitelnost obsažena a zjednodušuje její zneužití.

Pokud jsou informace o kritické zranitelnosti zveřejněny před jejím nahlášením včetně PoC („proof of concept exploit", tedy funkční ukázky zneužití zranitelnosti), zneužití zranitelnosti je primitivní nebo je zranitelnost aktivně zneužívána, musí být opravena co možná nejdříve či alespoň zveřejněna jiná opatření, které zneužití zranitelnosti minimalizují (tzv. workaround, např. vypnutí problematické části aplikace).

O.6 Všechny opravené zranitelnosti jsou uvedeny v souboru se změnami

Všechny zranitelnosti nahlášené i nalezené, např. interním penetračním testem či kontrolou kódu vývojářem, musí být uvedeny v souboru popisující provedené změny v jednotlivých verzích (typicky soubor CHANGELOG), který je součásti repozitáře.

Pro závažné zranitelnosti (CVSS 7.0 a vyšší) je přiřazen kód CVE. Kód CVE je uveden v souboru se změnami.

Kód CVE je globálně používaný unikátní identifikátor zranitelnosti spravovaný americkou neziskovou organizací MITRE. Výhodou je, že se podle tohoto kódu dá vyhledat o jakou zranitelnost se jedná a uživatelé aplikace nebo knihovny mohou tento identifikátor používat pro odkazování na konkrétní zranitelnost. Pro přiřazení CVE pro open-source projekty je možné využít formulář na cveform.mitre.org.

O.7 Účty vývojářů při autentizaci používají vícefaktorovou autentizaci

Všichni vývojáři, kteří mají práva:

- začleňovat nebo jinak měnit zdrojový kód v hlavní větvi repozitáře,
- vydávat nové verze,
- spravovat uživatelské účty,

používají vícefaktorovou autentizaci s nejméně dvěma různými typy faktorů pro přístup do systému správy kódu v případě, že je tento systém přístupný z internetu. Nejlépe, pokud je toto

nastavení možné vynutit nastavením politiky repozitáře nebo celého systému. Pokud je použit přístup přes kryptografický klíč, např. při přístupu pomocí Secure Shell (SSH), tento klíč využívá odolné kryptografické prostředky (viz C.2). Pokud je to možné, doporučujeme tento klíč mít uložen na hardwarovém kryptografickém modulu (např. HSM).

Kryptografické podepisování jednotlivých změn kódu je doporučeno.

O.8 Účty vývojářů jsou svázány s pracovní e-mailovou adresou

Všechny účty vývojářů v systému správce kódu, kteří mají práva začleňovat nebo jinak měnit zdrojový kód v hlavní větvi repozitáře,

- vydávat nové verze,
- spravovat uživatelské účty,

jsou svázány s pracovní e-mailovou adresou vývojáře. Je možné taktéž využít jiného poskytovatele identit, pokud je tato identita svázána s pracovní e-mailovou schránkou.

O.9 Dokumentace je součástí repozitáře

Dokumentace (popisující alespoň bezpečnostní mechanismy a jejich použití, v případě serverových aplikací také informace a požadavky k nasazení) je součástí repozitáře a verzována spolu s kódem.

Pokud aplikace zpřístupňuje aplikační programové rozhraní (API), je toto rozhraní dokumentováno ve strojově zpracovatelném formátu dle některého z otevřených standardů (např. OpenAPI).

O.10 Pro knihovny: Zranitelné verze knihoven jsou označeny

Pokud je knihovna zveřejněna ve veřejném správci balíčků, verze obsahující zranitelnosti jsou označeny jako zranitelné (pokud to správce balíčků umožňuje, obvyklý název této funkcionality je yanked) nebo jsou z něj odstraněny.

O.11 Neudržované aplikace a knihovny jsou označeny

Pokud je aplikace nebo knihovna ze strany organizace již dále neudržována, a tedy organizace již nebude reagovat na nahlášené zranitelnosti (např. v případě, kdy organizace tuto aplikaci nebo knihovnu už dále nevyužívá), je repozitář se zdrojovým kódem označen jako neudržovaný (např. funkcí správce kódu, v popisu repozitáře nebo v souboru README) a zároveň je tato informace uvedena i v souboru SECURITY (viz O.3).

Pokud se jedná o knihovnu zveřejněnou ve správci balíčků, je knihovna takto označena i v tomto správci.

O.12 Pro aplikace: Výchozí konfigurace je restriktivní

Konfigurace aplikace je nastavena tak, aby po instalaci omezovala potenciálně nebezpečné funkce a omezovala přístup k aplikaci – tedy například v případě webové aplikace naslouchala pouze na lokálním rozhraní (localhost) nebo neumožňovala připojení na nezabezpečené servery.

Pokud aplikace při instalaci vytváří uživatelské účty, musí mít vygenerovány náhodné heslo, u kterého je vynucena změna po prvním přihlášení do systému.

3 Správa a tvorba kódu

S.1 Zdrojový kód je verzován (VCS) a zveřejněn v otevřeném repozitáři

Zdrojový kód je verzován v otevřeném repozitáři, ke kterému má přístup široká veřejnost. Jednotlivé změny („commity") jsou logicky strukturované, aby mohly být zkoumány jak uživateli aplikace nebo knihovny, tak bezpečnostními výzkumníky. Změny, který mění bezpečnost systému (např. použitý kryptografický prostředek, opravení bezpečnostní chyby, změna způsobu autentizace apod.) jsou popsány buď v popisu změny („commit message") nebo v komentáři u zdrojového kódu.

S.2 Pro vývoj se používají oddělené větve, které se následně slučují do hlavní vývojové větve

Doporučujeme zakázat přímé vkládání změn do hlavní vývojové větve („zakázat commitování do masteru") a taktéž změny historie hlavní vývojové větve („force push do masteru"). Vývoj probíhá v oddělených větvích, které se následně začleňují do hlavní větve pomocí požadavků na začlenění do hlavní větve (v terminologii správců kódu Pull request nebo Merge request). Do hlavní vývojové větve je umožněn kód začlenit, jen pokud byla kontinuální integrace (viz sekce CI) úspěšná („zakázat merge, pokud neprojde CI").

S.3 Je prováděna kontrola změn kódu

V případě knihoven nebo aplikací, na jejichž vývoji se podílí více vývojářů, požadavky na začlenění kódu, které mohou mít vliv na bezpečnost systému, podléhají kontrolou a schválení jiným vývojářem.

Pokud je umožněno do kódu přispívat vývojářům z řad veřejnosti, v případě požadavku na začlenění kódu od neznámého vývojáře je tato kontrola prováděna vždy.

S.4 Repozitář neobsahuje binární spustitelné soubory nebo kompilované kódy

Aplikace, knihovna ani proces jejich sestavení by neměl záviset na spustitelných souborech (včetně zkompilovaných knihoven) umístěných v repozitáři. Pokud je to nezbytné, je k tomuto zkompilovaném souboru alespoň přidán popis, odkud byl soubor získán nebo jak je možné jej sestavit.

S.5 U dynamicky typovaných jazyků je využíváno striktní typování

U některých dynamicky typovaných jazyků může představovat dynamické typování nepředvídatelné chování a vést k bezpečnostním zranitelnostem. Pokud to využitý jazyk umožňuje, všechny funkce mají definované typy vstupních a výstupních parametrů. Některé programovací jazyky umožňují typovou kontrolu provádět za běhu nebo externím nástrojem v rámci CI.

4 Použité knihovny

Moderní open-source software je typický tím, že využívá velké množství knihoven s otevřeným zdrojovým kódem vyvíjených třetími stranami. Použití těchto knihoven šetří náklady na vývoj (není potřeba implementovat funkce, které již implementoval někdo jiný a svoji práci zveřejnil) a taktéž může zvyšovat bezpečnost (knihovna mohla být prověřena větším množstvím uživatelů).

Zároveň ale začleňování kódu třetích stran přináší riziko v možné zranitelnosti v kódu těchto otevřených knihoven. K repozitáři s kódem může získat přístup útočník např. převzetím tzv. „hacknutím" účtu správce, načež k legitimnímu kódu přidá škodlivý kód. Taktéž u populární knihovny existuje větší pravděpodobnost, že pro zranitelnost bude existovat zveřejněný způsob jejího zneužití a že bude potenciálními útočníky aktivně vyhledávána.

Následující doporučení nezakazuje použití knihoven třetích stran, ale definuje pravidla, jejichž dodržení by mělo vést k minimalizaci existence zranitelnosti v použité knihovně a jejímu rychlému vyřešení.

D.1 Aplikace a knihovny využívají udržované závislosti

Pokud je využita externí závislost, musí být použity pouze knihovny a jejich verze, které jsou udržované, aby bylo sníženo riziko použití knihovny, která obsahuje zranitelnosti, které nikdo nebude řešit. Pokud aplikace závisí na neudržované knihovně, měla by být nahrazena novější, podporovanou verzí nebo její udržovanou alternativou.

Znaky udržované knihovny (nemusí být splněny všechny):

- za poslední rok byla vydána alespoň jedna nová verze knihovny,
- do zdrojového kódu knihovny přispívá více než jeden vývojář,
- tvůrci knihovny reagují na otevřená issues a žádosti o začlenění kódu,
- tvůrce knihovny deklaruje řešení nahlášených zranitelností,
- bylo vydáno aspoň pět předchozích verzí.

D.2 Preferovány jsou knihovny s bezpečným API

Bezpečné API znamená, že knihovní funkce jsou automaticky ošetřeny před nebezpečným vstupem či výstupem. Potenciálně nebezpečné funkcionality jsou ve výchozím nastavení zakázány a vývojář je musí explicitně povolit (zjednodušeně řečeno – bezpečné použití je jednodušší než nebezpečné). Použití ověřené knihovny s bezpečným API by mělo být i preferováno před vlastní implementací.

Příklady knihoven s bezpečným API:

- Knihovna pro práci s databází automaticky escapuje vstupní parametry.
- Šablonovací systém automaticky escapuje výstupní řetězce.

- Knihovny pro syntaktickou analýzu standardizovaných formátů (XML, JSON, apod.) mají ošetřeny nevalidní vstupy, které nevedou k neplatnému čtení paměti.

Příklady knihoven s nebezpečným API:

- Serializační knihovny, které při deserializaci umožňují spouštění kódu. 1
- Knihovna pro práci s databází, u které vývojář musí escapovat vkládaná data voláním další funkce.

D.3 Preferovány jsou knihovny, jejichž autoři „dbají na bezpečnost"

Při výběru knihovny by měly být preferovány ty, jejichž autoři dbají na bezpečnost vývoje.

Znaky knihovny, u nichž autoři dbají na bezpečnost:

- Je prováděna kontinuální integrace (CI),
- repozitář obsahuje soubor SECURITY,
- veřejný seznam nahlášených Issues neobsahuje neopravené nahlášené zranitelnosti,
- soubor CHANGELOG obsahuje nalezené bezpečnostní zranitelnosti,
- oprava nahlášených bezpečnostních chyb trvá méně než 30 dní,
- nalezené zranitelnosti nejsou triviálního charakteru (případně nejsou obvyklé),
- v rámci hledání zranitelností autoři používají technik fuzzingu.

D.4 Pro aplikace: Aplikace při definování závislostí používají „lock file"

Aplikace specifikuje přesné verze závislostí, se kterými byla testována, pokud to daný správce závislostí umožňuje. Cílem je zabránit využití novější a potenciálně podvržené verze závislostí, kdy útočník získá přístup k účtu správce knihovny a nahradí ji podvrženou verzí.

V případě webové aplikace, která načítá knihovny z externích serverů (CDN) je k tomuto účelu možné použít technologii SRI (Subresource Integrity) podporovanou v moderních webových prohlížečích.

D.5 Závislosti jsou stahovány z důvěryhodných úložišť

Při výběru správců závislostí jsou preferováni takoví, kteří kontrolují integritu stahovaného balíčku pomocí podpisu tvůrce aplikace nebo alespoň pomocí haše. Pokud správce závislostí kontrolu integrity neumožňuje, závislosti jsou výhradně stahovány pomocí zabezpečeného kanálu (HTTPS, SSH apod.).

Pokud je nutné využít nepublikovanou verzi knihovnu ze správce závislostí, je specifikován tag nebo identifikátor konkrétní změny („commitu").

5 Kontinuální integrace

Kontinuální integrace (continuous integration, CI) je nástroj, při kterém dochází k automatizovanému spouštění definovaných procesů (např. sestavení a testování) při každé změně kódu. Následující doporučení určují procesy, které by se v rámci kontinuální integrace měly provádět.

CI.1: Pro aplikace: Jsou kontrolovány verze použitých závislostí

Před každým sestavení aplikace je kontrolováno, zda použité závislosti neobsahují bezpečnostní zranitelnosti. V případě vážných zranitelností nesmí být sestavení provedeno.

V případě potřeby výjimky (např. uvedená zranitelnost není relevantní pro konkrétní použití), musí být tato výjimka zdokumentována.

CI.2: Nalezené zranitelnosti jsou testovány v rámci CI

Pokud je nalezená zranitelnost testovatelná, měl by být vytvořen test, který zaručí, že zranitelnost byla správně opravena a nebude opět obnovena v následujících verzích. Nejlépe by měl být test ověřující zranitelnost přidán v prvním commitu, CI by mělo ohlásit chybu a následně po přidání opravy zranitelnosti by měl být test úspěšný.

CI.3: Pro aplikace: Sestavení aplikace je plně automatizované

Pokud je aplikace vydávána ve formě balíčku, instalátoru, binární aplikace nebo obrazu kontejneru, toto sestavení je prováděno v rámci použitého CI. Předpis umožňující sestavení je součástí repozitáře.

CI.4: Je definován a vynucován standard pro zdrojový kód

Jakmile budou přijata rozhodnutí o použitých technologiích, musí být vytvořeny, udržovány a sděleny příslušné vývojové standardy a konvence, které podporují jak psaní bezpečného kódu, tak opětovné použití vestavěných bezpečnostních funkcí a schopností. Tento standard je součástí repozitáře a je kontrolován v rámci kontinuální integrace, nepovinně i v rámci vývojového prostředí.

Tento standard by měl mimo jiné obsahovat předpis (ne)povolených funkcí a kontrolu použitých funkcí dle použitého jazyka (Java, PHP, .NET), včetně doporučených ekvivalentů k těmto funkcím (některé jazyky používají nebezpečné funkce generované v rámci běhové prostředí a jsou zdrojem zranitelností).

CI.5: Jsou prováděny jednotkové nebo integrační testy v oblastech s vlivem na bezpečnost

V rámci kontinuální integrace jsou prováděny jednotkové (unit) nebo integrační testy v oblastech kódu, které můžou mít vliv na bezpečnost systému. Mezi tyto oblasti patří zejména:

- Autentizace do systému (autentizace se správným heslem, autentizace bez hesla, autentizace s neplatným heslem, autentizace zablokovaného uživatele atd.).

- Autorizace do jednotlivých částí systému (přístup do částí bez potřebného oprávnění).
- Použití kryptografických knihoven.

CI.6: Jsou prováděny automatizované bezpečnostní testy

V rámci kontinuální integrace je použit nástroj, který zajistí základní bezpečnostní testy. Typ nástroje je volen dle použitého jazyka, frameworku a typu knihovny nebo aplikace. Jedná se například o nástroje na statickou analýzu kódu, aktivní skener bezpečnostních zranitelností, v případě aplikace pracující s uživatelským vstupem fuzzer. Jelikož doba běhu některých nástrojů může být neúměrně dlouhá, je možné některé testy spouštět jen před plánovaným vydáním nové verze.

Typy bezpečnostních testů:

- Static Analysis Security Testing (SAST) – kontrola nedostatků zdrojového kódu nebo kompilovaného mezijazyka nebo binární komponenty. Hledá známé problematické vzory v kódu založené pouze na aplikační logice, nikoli na chování aplikace při jejím spuštění. SAST nepokrývá detekci zranitelností v business logice, problémů zavedených na více úrovních aplikace nebo třídy problémů vytvořených za běhu.
- Dynamic Analysis Security Testing (DAST) – testování předpřipravených útoků vůči plně běžící (kompilované) aplikaci s veškerou integrací potřebných komponent (podpora testování aplikace napsané i v jazyce nepodporovaném SAST nástrojem nebo v případech, kdy aplikace využívá externích volání webových služeb nebo JavaScriptových knihoven uložených mimo repozitář kódu).
- Fuzzing – mnohonásobné variabilní generování nebo mutace dat a jejich předání funkcím zajišťujících syntaktickou analýzu na úrovni aplikačních dat (síťové protokoly, souborové, IPC). Dobré pokrytí kontroly kódu, zejména pro jazyky C a C++.
- Software Composition Analysis (SCA) – zabývá se správou používání komponent s otevřeným zdrojovým kódem. Nástroje SCA provádějí automatické skenování kódové základny aplikace včetně souvisejících artefaktů, jako jsou kontejnery a registry, s cílem identifikovat všechny komponenty s otevřeným zdrojovým kódem, údaje o jejich souladu s licencemi a případné bezpečnostní zranitelnosti.

CI.7: Je prováděna kontrola tajných identifikátorů ve zdrojovém kódu

V rámci kontinuální integrace je použit nástroj, který detekuje, zda zdrojový kód neobsahuje známé soubory nebo řetězce, které obsahují přístupové údaje (např. hesla, tokeny pro přístup, soukromé SSH klíče, certifikáty se soukromým klíčem apod.) v angličtině označované jako „secrets".

Při pozitivním nálezu je nutné považovat tajný identifikátor za kompromitovaný a musí být změněn či revokován bez ohledu, jak dlouho byl zveřejněn. Změna historie repozitáře není dostačující.

CI.8: Pro aplikace: Vydané verze jsou podepsány

Vydané verze aplikací jsou kryptograficky podepsány, případně je alespoň zveřejněn haš (nejlépe SHA-256) výsledného souboru.

6 Kryptografické prostředky

Prakticky žádná aplikace se nevyhne používání kryptografických prostředků ať už při komunikaci s ostatními systémy nebo při komunikaci uživatele s aplikací. Jelikož tvorba a implementace kryptografických algoritmů je složitá oblast náchylná na jakékoliv drobné chyby a v softwaru s otevřeným zdrojovým kódem je pro případného útočníka snadnější tyto chyby nacházet, doporučujeme se vyvarovat návrhu či případně implementaci vlastních algoritmů, a naopak využívat algoritmy z ověřených kryptografických knihoven. Tyto knihovny obvykle prochází pravidelnou kontrolou bezpečnostních výzkumníků a nalezené zranitelnosti jsou rychle opravovány.

C.1 Jsou využívány ověřené kryptografické knihovny

Naprogramovat kryptografické algoritmy tak, aby byly odolné proti různým typům útoků, je velmi složité. Proto v aplikacích a knihovnách jsou primárně využívány kryptografické funkce dostupné v rámci základní knihovny použitého jazyka, frameworku nebo systému, případně jiné ověřené knihovny (viz L.1).

Pokud je nutné vytvářet implementace vlastních algoritmů, v komentáři u kódu je uvedeno, z jakého důvodu byla zvolena vlastní implementace.

C.2 Jsou využívány odolné kryptografické prostředky

V případě, že aplikace nebo knihovna přímo definuje použité kryptografické prostředky, musí využívat jen ty aktuálně odolné.

Pro výběr vhodných kryptografických algoritmů je možné využít Doporučení v oblasti kryptografických prostředků vydávaných NÚKIB. Tento dokument rozlišuje dvě kategorie algoritmů: schválené, které jsou bezpečné alespoň ve střednědobém horizontu, a dosluhující, které by se měly přestat používat po roce 2023 a nezavádět se v nových systémech.

Jiný algoritmus může být použit jen v nezbytných případech (např. kvůli zpětné kompatibilitě nebo komunikaci s jiným systémem nepodporující odolný algoritmus).

C.3 Při použití protokolu TLS je podporována alespoň verze 1.2

V případě použití kryptografického protokolu TLS je podporována alespoň verze 1.2 a to jak při příchozím, tak odchozím spojení. Starší verze můžou být použity jen v nezbytných případech.

C.4 Je prováděno ověřování použitého certifikátu

Při navazování zabezpečeného odchozího spojení, kde se používá k ověření protistrany kryptografický certifikát, je u tohoto certifikátu kontrolováno alespoň:

- zda je podepsán důvěryhodnou certifikační autoritou,
- zda není vypršený nebo naopak před dobou platnosti,
- zda Common Name nebo Alternative DNS Names odpovídá použité doméně.

Tyto kontroly jsou u většiny knihoven určených pro navazování zabezpečené komunikace zapnuty ve výchozím nastavení, vývojář by je proto měl vypínat pouze v odůvodněných případech pro konkrétní spojení. Pro ověření kontrol certifikátů je možné použít například službu badssl. Pokud knihovna tyto kontroly nepodporuje, je potřeba ji vyměnit za bezpečnější nebo kontroly provádět dodatečně ve vlastním kódu.

Je taktéž možné použít jiný způsob ověření certifikátu, který zaručuje obdobnou nebo vyšší bezpečnost (např. pomocí protokolu DANE a DNSSEC nebo uvedení otisku certifikátu v konfiguračním souboru).

C.5 Aplikace používá systémové úložiště důvěryhodných certifikátů

Při kontrole, zda je certifikát podepsán důvěryhodnou certifikační autoritou je využíváno úložiště certifikátů operačního systému. Některé knihovny využívají svůj vlastní seznam důvěryhodných certifikačních autorit, to ale vede k jejich obtížné správě. V rámci aplikace je možné přidat další certifikační autoritu (např. interní používanou v rámci organizace).

C.6 Uložená hesla jsou odolná proti offline útokům

Pokud aplikace pracuje s uživatelskými hesly nebo jinými autentizačními údaji a ukládá je do databáze či do souboru, uložené údaje musí být chráněny proti offline útokům (tzn. musí být uloženy v takové formě, u které je výpočetně náročné i se znalostí uložených údajů získat údaje původní).

V případě, že není potřeba pracovat s originálním údajem, doporučujeme k jejich zahašování využít jeden z následujících algoritmů (v pořadí od nejvhodnějšího):

- Argon2 (nejlépe ve verzi „id") – využívá hašovací algoritmus BLAKE2, který je schválen
- scrypt – využívá hašovací algoritmus SHA-256, který je schválen
- bcrypt – využívá blokovou šifru blowfish, která je dosluhující
- PBKDF2 – umožňuje volbu hašovací algoritmu, doporučujeme využití schváleného hašovací algoritmu dle C.2

Sůl („salt") musí být generována pomocí k tomu určenému algoritmu (viz C.8), doporučujeme zvolit sůl minimálně o velikosti 64 bitů (lépe 128 bitů). Pokud je možné zvolit výpočetní náročnost algoritmu, doporučujeme ji nastavit tak, aby výpočet trval minimálně 100 ms (lépe 500 ms) a využil minimálně 1 MB paměti.

C.7 Porovnání hašů uložených hesel je odolné na časovou analýzu

Běžné porovnání řetězců je náchylné na časovou analýzu („timing attack") – z důvodu optimalizace je porovnávání ukončeno při nalezení první neshody a doba porovnání je tedy závislá na uživatelském vstupu. Případný útočník může tímto způsobem odhadnout použité heslo.

Odolné funkce jsou tedy takové, kdy porovnání trvá vždy stejnou dobu bez ohledu na použitý vstup. Není tedy možné využít běžné porovnání (ve většině programovacích jazyků pomocí operátoru ==), které porovnání ukončí po nalezení první neshody.

C.8 Ke generování náhodných tokenů jsou použity kryptograficky bezpečné pseudonáhodné generátory

Pokud je v aplikaci nebo knihovně generován náhodný token, je pro toto generování použit kryptograficky bezpečný pseudonáhodný generátor – tyto funkce jsou označované jako CSPRNG (cryptographically secure pseudorandom number generator) nebo CPRNG (cryptographic pseudorandom number generator). Mezi náhodné tokeny patří například:

- přístupový klíč,
- session ID,
- kryptografická nonce (např. inicializační vektor nebo sůl při ukládání hesel – viz C.6),
- identifikátor zaslaný na e-mail pro obnovu hesla,
- prvotní heslo uživatele.

C.9 Tajné identifikátory jsou odstraňovány z paměti

Datové struktury obsahující citlivá data (heslo, privátní klíč, session ID apod.) jsou v paměti udržovány pouze po nezbytně nutnou dobu. Pokud již dále nejsou potřeba, jsou přepsány pseudonáhodnou sekvencí, pokud to použitý programovací jazyk umožňuje.

7 Záznamy o událostech

L.1 Pro knihovny: Knihovny využívají standardní rozhraní pro logování

Knihovny využívají standardní rozhraní pro zaznamenávání událostí dle použitého jazyka, případně dle použitého frameworku. Přímé logování do souboru či na standardní chybový výstup („stderr“) je možné jen, pokud není možné standardní rozhraní použít či neexistuje.

L.2 Pro aplikace: Aplikace umožňuje logování důležitých akcí

Aplikace zaznamenává důležité události provedené administrátory, uživateli nebo samotným systémem. Pro inspiraci je možné využít § 22 vyhlášky č. 82/2018 Sb., který definuje typy událostí, o kterých by měl být proveden záznam:

- přihlašování a odhlašování ke všem účtům, a to včetně neúspěšných pokusů,
- činnosti provedené administrátory,
- úspěšné i neúspěšné manipulace s účty, oprávněními a právy,
- neprovedení činnosti v důsledku nedostatku přístupových práv a oprávnění,
- činnosti uživatelů, které mohou mít vliv na bezpečnost informačního a komunikačního systému,
- zahájení a ukončení činnosti technických aktiv,
- kritické i chybové hlášení technických aktiv a
- přístupů k záznamům o událostech, pokusy o manipulaci se záznamy o událostech a změny nastavení nástrojů pro zaznamenávání událostí.

L.3 Pro aplikace: Aplikace podporuje napojení na centrální log management

Aplikace musí umožňovat napojení na centrální log management zasíláním strukturovaných informací vhodným obecně podporovaným standardem (např. syslog). Každý log záznam obsahuje minimálně následující informace (dle vyhlášky):

- datum a čas včetně specifikace časového pásma,
- typ činnosti (viz L.2),
- identifikaci technického aktiva, které činnost zaznamenalo (např. hostname a název aplikace),
- jednoznačnou identifikaci účtu, pod kterým byla činnost provedena,
- jednoznačnou síťovou identifikaci zařízení původce (např. IP adresa) a
- úspěšnost nebo neúspěšnost činnosti.

V případě webové aplikace je možné zvolit, z jakého zdroje je získávána IP adresa původce, tak aby nebyla zaznamenána pouze IP adresa reverzní proxy.

L.4 Nejsou zaznamenávány tajné identifikátory

V žádné úrovni logování nejsou do log záznamů vkládány tajné identifikátory (hesla, přístupové tokeny, privátní klíče apod.) – buď jsou z logování vynechány nebo nahrazeny bezvýznamovou hodnotou. Pokud je pro vývoj nutné mít k těmto tajným identifikátorům přístup, je potřeba toto logování povolit zvláštním parametrem (např. proměnnou prostředí). Taktéž je potřeba omezit logování osobních údajů, pokud nejsou nezbytné k zajištění bezpečnosti systému. Omezení logování tajných identifikátorů a osobních údajů se vztahuje jak na logy z aplikace, tak např. na přístupové logy z webserveru. Tyto údaje by tím pádem neměly být ani součástí URL adres.

L.5 Výjimky jsou řízeny

Systém musí podporovat řízení výjimek, kdy výjimkou se myslí libovolná chyba nebo neočekávané chování, které se vyskytne během vykonávání programu a je následně zpracováno a zároveň nedojde k neřízenému selhání běhu.

V případě uživatelské aplikace bude při vzniku chyby běhu programu zobrazeno dialogové okno s identifikátorem chyby mající vazbu na log události aplikace, pod kterým je situace následně v lozích dohledatelná, přičemž musí existovat oddělení uživatelských hlášení od technických. Uživatelská hlášení nesmí obsahovat technické detaily (jako např. traceback), ale jen identifikátor, který odkazuje na jeho popis mimo systém. Opakované a známé chyby je vhodné opatřit kódem a smysluplným popisem, aby je chápal běžný uživatel systému. Volitelně jsou tyto chyby odesílány do centrálního systému pro správu výjimek.

V případě systému, který neinteraguje s uživatelem, jsou výjimky logovány či zaslány do centrálního systému pro správu výjimek.

8 Relační databáze

Následující body jsou relevantní, pouze pokud aplikace využívá relační databázi pro ukládání dat a návrh databázového schématu je součástí aplikace. Tato pravidla cílí na zajištění integrity ukládaných dat a jednodušší přenositelnost dat do jiných systémů.

D.1 Jsou využívány primární klíče

Primární klíč je hodnota, která jednoznačně identifikuje uloženou entitu v databázi. Při smazání záznamu nesmí být tento identifikátor znovu použit a musí být na úrovní databáze zajištěno, že se v databázové tabulce vyskytuje pouze jednou.

D.2 Jsou používány cizí klíče při vazbě na číselníkové hodnoty anebo na jiné entity

Při vazbě jedné entity na druhou jsou využívány cizí klíče, aby byla zajištěna integrita záznamů, tedy že jedna entita nebude odkazovat na neexistující entitu.

D.3 V rámci databáze jsou kontrolovány hodnoty parametrů („constraints")

Pokud má určitý sloupec omezený počet hodnot, kterých může nabývat, jsou tyto možné hodnoty omezeny přímo na úrovni databáze.

D.4 Pokud databázový sloupec předpokládá unikátní hodnotu, je využit unikátní index

V případě, že entita obsahuje hodnotu, která musí být unikátní, musí být využit unikátní index.

Upozornění: některé hodnoty, které by měly být unikátní, unikátní být nemusí (např. rodné číslo). V takovém případě na tuto kolizi musí být systém připraven.

D.5 Při vkládání dat jsou využívány transakce

Pokud jsou v jedné transakci měněny nebo přidávány hodnoty do více databázových tabulek, jsou využity transakce na úrovni databáze, které zajistí integritu ukládaných dat.

D.6 Databázové schéma je komentováno

Tabulky a sloupce jsou vhodně komentovány, aby byl jasný jejich význam a vazby na jiné tabulky.

9 Podmínky využití informací

Využití poskytnutých informací probíhá v souladu s metodikou Traffic Light Protocol (dostupná na webových stránkách https://www.nukib.cz/cs/infoservis/doporuceni/1593-doporuceni-k-pouzivani-protokolu-tlp-ke-sdileni-chranenych-informaci/). Informace je označena příznakem, který stanoví podmínky použití informace. Jsou stanoveny následující příznaky s uvedením charakteru informace a podmínkami jejich použití:

Barva	Podmínky použití
Červená TLP: RED	Informace nemůže být poskytnuta jiné osobě než té, které byla informace určena, nebudou-li výslovně stanoveny další osoby, kterým lze takovou informaci poskytnout. V případě, že příjemce považuje za důležité informaci poskytnout dalším subjektům, lze tak učinit pouze se souhlasem původce informace.
Oranžová TLP: AMBER	Informace může být sdílena v rámci organizace, které byla informace poskytnuta. Dále může být poskytnuta pouze těm partnerům, kteří splňují need-to-know a jejichž informování je důležité pro vyřešení problému či hrozby uvedené v informaci. Jiným osobám, než výše uvedeným, nesmí být informace poskytnuta. Původce informace může rozsah sdílení dále omezit.
Zelená TLP: GREEN	Informace může být sdílená v rámci organizace příjemce a případně také s dalšími partnerskými subjekty příjemce, avšak nikoli skrze veřejně dostupné kanály; příjemce musí při předání zajistit důvěrnost komunikace.
Bílá TLP: WHITE	Informace může být dále poskytována a šířena bez omezení. Případné omezení na základě práva duševního vlastnictví původce a/nebo příjemce či třetích stran nejsou tímto ustanovením dotčena.

Verze dokumentu			
datum	verze	změněno	popis změny
5. dubna 2022	1.0	Vládní CERT	Vytvoření dokumentu

10 Příloha: Příklady souborů SECURITY a security.txt

Příklad souboru SECURITY:

```
## Reporting security vulnerabilities

Reporting security vulnerabilities is of great importance for us, as
this project is used in critical infrastructure.

In the case of a security vulnerability report, we ask the reporter to
send it directly to …, if possible encrypted with the following PGP
key: **…**. We usually fix reported and confirmed security
vulnerabilities in less than 48 hours, followed by a software release
containing the fixes within the following days.

If you report security vulnerabilities, do not forget to tell us if
and how you want to be acknowledged and if you already requested
CVE(s). Otherwise, we will request the CVE(s) directly.

## Hlášení bezpečnostních zranitelností

Nahlášení bezpečnostní zranitelnosti je pro nás velmi důležité,
protože tento projekt je využíván v rámci kritické infrastruktury.

V případě bezpečnostní zranitelnosti nás prosím přímo kontaktujte …,
pokud možno zašifrovaným e-mail s PGP klíčem **…**. Obvykle provedeme
opravu a potvrdíme zranitelnost za méně než 48 hodin, v další dny
vydáme novou verzi obsahující opravu této zranitelnosti.

Pokud hlásíte bezpečnostní zranitelnost, nezapomeňte uvést, jak chcete
být zveřejněni v changelogu a zda jste již požádali o přidělení CVE.
Pokud ne, zažádáme o přidělení CVE za vás.
```

Příklad souboru security.txt:

```
Contact: mailto:cert@nukib.cz
Expires: 2023-01-01T00:00:00.000Z
Encryption: https://www.nukib.cz/download/kontakty/cert_pub.asc
Preferred-Languages: cs, sk, en
Hiring: https://kariera.nukib.cz/
```