ELEKTROTEHNIŠKI VESTNIK 92(5): 250–260, 2025
IZVIRNI ZNANSTVENI ČLANEK

Metoda varnega predogleda sumljivih priponk PDF z zaklenjeno pretvorbo v slikovni zapis in e-poštnim agentom kot vmesnikom za uporabnika

Žiga Rojec

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: ziga.rojec@fe.uni-lj.si

Povzetek. Datoteke PDF so lahko nosilke zlonamerne kode, ki v obliki priponk dosegajo uporabnikov e-poštni predal. Prispevek predlaga novo metodo tehnične rešitve za varen, hiter in enostaven ogled datotek PDF dvomljivega izvora. Z uporabo statično povezane knjižnice Poppler (Linux) napravimo enosmerno lokalno pretvorbo vsebine PDF v slikovni zapis (JPG). Proces izvajanja pretvorbe izvedemo znotraj peskovnika Seccomp v depriviligiranem vsebniku Docker, ki varuje zaledni operacijski sistem pred morebitnim izvajanjem škodljive kode med pretvorbo dokumenta. Takšna zasnova zmanjšuje napadalno površino, povezano z dinamično interpretacijo strukture PDF, in je primerna za integracijo v okolja z omejenimi varnostnimi privilegiji. Pretvorba se izvede izjemno hitro, saj se izognemo zahtevam po polni virtualizaciji sistema, ki bi jo sicer morali uporabiti za doseganje podobne ravni varnosti. Poleg samostojne izvedbe v obliki lokalnega orodja CLI članek predlaga arhitekturo oblačne mikrostoritve z agentom za elektronsko pošto, ki omogoča razumljiv in nezahteven uporabniški vmesnik na ravni enostavnega pošiljanja sporočil elektronske pošte agentu sistema. Prispevek vključuje tehnični opis izvedbe, oceno učinkovitosti in primerjavo s klasičnimi virtualizacijskimi pristopi.

Ključne besede: kibernetska varnost, peskovnik, odpornost, elektronska pošta, zlonamerna koda, Seccomp, Docker

A Method for Secure Preview of Suspicious PDF Attachments Using Locked-Down Image Conversion and an Email Agent as the User Interface

PDF files can act as carriers of malicious code, reaching users' email inboxes in the form of attachments. This paper proposes a novel method for a technical solution enabling the secure, fast, and simple viewing of PDF files from untrusted sources that may potentially contain malicious code. Using the statically linked Poppler library, a one-way local conversion of the PDF content into an image format (JPG) is performed. The conversion process is executed within a Seccomp sandbox located within an unprivileged Docker container, which protects the underlying operating system from the potential execution of malicious code during the document conversion. This design reduces the attack surface associated with the dynamic interpretation of the PDF structure and is suitable for integration into environments with limited security privileges. The conversion is exceptionally fast, as it eliminates the need for full system virtualization that would otherwise be required to achieve a similar level of security. In addition to a standalone implementation as a local CLI tool, the paper proposes an architecture for a cloud-based microservice with an email agent, which provides an intuitive and user-friendly interface at the level of simply sending an email message to the system agent. The paper includes a technical description of the implementation, an efficiency evaluation, and a comparison with classical virtualization-based approaches.

Prejet 10. julij, 2025
Odobren 13. november, 2025

1 UVOD

1.1 Položaj formata PDF v digitalnem ekosistemu

PDF (Portable Document Format) je med najbolj razširjenimi formati za izmenjavo zaključenih dokumentov [7], [36]. Vse od standardizacije leta 2008 (ISO 32000-1) je *de facto* valuta za izmenjavo zaključenih dokumentov v znanosti, javni upravi in gospodarstvu [4]. Njegove ključne prednosti so:

- **prenosljivost** – dokument na vseh napravah ohrani identičen videz;
- **pestrost funkcij** – vgradnja slik, zvoka, videa, certifikatov in JavaScript-a [2], celo 3D-modelov [19];
- **široka združljivost** – na stotine odprtokodnih in komercialnih pregledovalnikov [21], [23], [26], [15], [34], [22], [28].

Nedavna poročila ocenjujejo, da globalno več kot 60 % vseh digitalnih dokumentov kroži v obliki PDF in da letno število celo raste [7].

1.2 Skrita cena univerzalnosti

Struktura dokumenta PDF po standardu ISO 32000 lahko vsebuje *akcije*, ki jih pregledovalnik izvede samodejno (npr. ob odpiranju dokumenta) ali na podlagi uporabnikove interakcije (klik, premik miške). Posebno problematična so ukazna polja, ki lahko prožijo:

- samodejno povezavo na oddaljen URL,

- prenos in zagon vdelanega JavaScript-a,
- branje in zapis lokalnih datotek preko starejših vtičnikov [32], [4].

Ker so te zmožnosti del standarda, jih morajo pregledovalniki implementirati, to pa lahko poveča t. i. napadalno površino. Zlasti problematično je avtomatično povezovanje na URL-povezave, ki se zgodi ob odprtju datoteke. Mehanizmi za detekcijo (protivirusni programi) vgrajene škodljive kode ne zaznajo, ob odprtju datoteke pa se ta lahko naloži s tretjega strežnika, se zažene in povzroči škodo uporabniku.

1.3 Razredi grožnj

Müller idr. [32] tveganja razvrščajo v štiri osrednje razrede:

1) **DoS** ("Denial-of-Service") – zavrnitev storitve; procesor in pomnilnik se zapolnita s kompleksnimi objekti, kar zamrzne aplikacijo;
2) **kraja podatkov** – pridobivanje poverilnic z obrazcev ali datotečnega sistema uporabnika;
3) **sprememba sistema** – pisanje nezaželenih datotek, sprememba registrskih ključev;
4) **oddaljeni zagon kode** (RCE) – izvrševanje vdelanih skript ali dinamično nalaganje binarnih modulov.

Dodatni razred predstavljajo t. i. *shadow attacks* – manipulacija vsebine po digitalnem podpisu – a ti ne sodijo v tematiko tega prispevka [30], [24].

1.4 Psihologija zaupanja uporabnikov

Zgodovina interneta je uporabnike naučila nezaupanja in previdnosti do priponk ZIP in dokumentov Office z makri. Nasprotno pa besedilne datoteke PDF večina označuje kot *relativno varne* [20]. Veliko zaupanje skupaj z dokazano ranljivostjo formata ustvarja funkcionalno vabo, ki jo lahko izrabijo napadalci. Po podatkih podjetij za internetno varnost je 22 % vseh škodljivih priponk v vektorskem kanalu e-pošte prav PDF [13]. Zaradi relativno velikega zaupanja v varnost formata in njegovih dokazanih varnostnih tveganj torej obstaja zelo velika verjetnost uporabe tega vektorja za izvedbo sofisticiranih napadov na uporabnika spleta, elektronske pošte ali druge informacijske tehnologije.

1.5 Živahno raziskovalno področje

Raziskovanje in reševanje problema varnosti PDF-datotek je živo akademsko področje. Dosedanji prispevki segajo od statične analize bajtnih tokov [27] do uporabe konvolucijskih nevronskih mrež za klasifikacijo s predučenjem [16], [37], [11] in preostalih metod strojnega učenja [38]. Kljub bogati literaturi so ključni izzivi t. i. *zero-day exploits* (škodljivi postopki, ki kot prvi izrabljajo drugim še neodkrito ranljivost), pobeg iz peskovnikov in uporabniška ergonomija, ki odpirajo nova znanstvena vprašanja.

1.6 Obstoječa zaščita: kje se zatakne?

1.6.1 Protivirusne zaščite: Uporabniki osebnih računalnikov se proti grožnjam zlonamerne kode lahko dobro zaščitijo z uporabo protivirusnih in *anti-malware* programov. Programi te vrste večinoma pregledujejo datoteke na uporabnikovem disku in njihove binarne prstne odtise primerjajo z vpisi v podatkovno bazo znanih škodljivih programov oz. virusov. Ob morebitnem ujemanju nato zaščitijo sistem tako, da škodljivo datoteko odstranijo iz sistema oz. uporabniku onemogočijo njegovo uporabo. Težava se pojavi, ko uporabnik prejme škodljivo datoteko, ki je protivirusna podjetja v preteklosti še niso zaznala pri dovolj uporabnikih, in zato še ni vključena v bazo protivirusnega programa. Če je škodljiva koda tehnološko izpopolnjena in usmerjena na točno enega uporabnika, pa je protivirusni program pri odkrivanju zlonamerne kode lahko zelo neuspešen.

Uporaba zaščitnih programov je pogosto povezana z dragimi licencami ter vzdrževanjem posodobljenega sistema. Poleg tega pa zaščitnih programov tipično ne poganjamo na napravah, kot so mobilni telefoni, tablice, računalniki z operacijskim sistemom Linux ali MacOS. Težje jih je posodabljati tudi na starejših sistemih, ki jih poganjajo operacijski sistemi Windows 10, Win XP in starejši, ki jih kljub upokojitvi proizvajalca uporablja več kot polovica uporabnikov osebnih računalnikov [3].

1.6.2 Strežniške karantene: Večji ponudniki storitev elektronske pošte (npr. Microsoft, Google) uporabljajo avtomatizirane karantenske mehanizme, ki so sposobni že na ravni strežnika oceniti varnost posredovane vsebine, še preden ta doseže naslovnika [31], [40]. Omenjeni sistemi imajo lahko določene pomanjkljivosti: poleg vsiljene pošte in škodljive vsebine pogosto blokirajo tudi legitimne pošiljke. Z delovanjem prav tako nižajo hitrost prenosa pošte – pogosto se poročila o sporočilih v karanteni pošiljajo enkrat na dan ali celo redkeje, zato uporabnik lahko spregleda pomembna sporočila, ki so mogoče legitimna, pa jih je programska oprema označila za potencialno škodljiva.

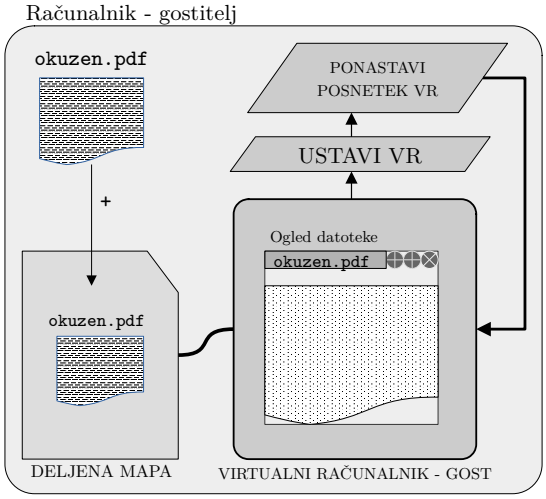
Karantene v oblaku ponujajo tudi drugi ponudniki oblačnih storitev. Uporabnik lahko sumljive datoteke enostavno odvrže (*drag-and-drop*) v spletno storitev, ki datoteko pregleda s protivirusnim programom [39], [17] in poda poročilo. Oblačne storitve lahko podajo tudi podrobnejšo analizo o obnašanju datoteke [9], [25], [14]. Slednja orodja so lahko zelo uporabna za izkušene računalniške strokovnjake, manj pa za vsakdanjo rabo običajnih uporabnikov. Opomniti moramo, da imajo brezplačni računi omenjenih ponudnikov peskovnikov v oblaku model, po katerem postanejo poslane datoteke *javne*.

1.6.3 Uporaba virtualizacije: Za varen in popolnoma zaseben vpogled v vsebino potencialno škodljive datoteke (ali zagon programa, ki mu ne zaupamo) lahko uporabnik uporabi preverjeno metodo izolacije v *virtual-*

nem stroju (angl. *virtual machine* ali *VM*). Na računalnik namestimo program za virtualizacijo operacijskega sistema (npr. VirtualBox, VMware Workstation, Parallels ali Hyper-V) [35]. Nato, kot opisano v sliki 1:

1) v VM namestimo svež operacijski sistem in pregledovalnik PDF,*
2) ustvarimo *posnetek stanja* (*snapshot*) neposredno po namestitvi,
3) konfiguriramo deljeno mapo gostitelj–gost za prenos dokumentov ter *onemogočimo* internetno povezavo in skupno odložišče,
4) kopiramo sumljivo priponko v deljeno mapo,
5) zaženemo VM in v izoliranem okolju odpremo PDF,
6) po ogledu zapremo VM in povrnemo stanje na prvotni *snapshot*.

Metoda je tehnično učinkovita, a zahteva čas, sistemsko znanje in discipliniran postopek; napake (npr. nenamerno odprtje datoteke med nameščanjem v deljeno mapo, nenamerno omogočen internet, pozabljen *snapshot*) lahko izničijo varnostne koristi. V razdelku *Rezultati in diskusija* ta postopek kvalitativno primerjamo z našim predlaganim pristopom.



Slika 1 Uporaba virtualnega računalnika za varen ogled vsebine PDF.

1.7 Uporabnikova dilema

V nasprotju z uporabniki v tehnično in informacijsko dobro opremljenih korporacijah ali javnih ustanovah pa večina manjših organizacij in samostojnih uporabnikov e-pošte pogosto nima možnosti uporabe naprednih karantenskih sistemov, ki bi zajezile tveganje prenašanja zlonamernih datotek v obliki priponk ali drugače. Veliko uporabnikov e-pošte se zaveda nevarnosti, pogosto pa nimajo ali ne poznajo načina varnega preverjanja vsebine. To uporabnike spravlja v dilemo – dokument odpreti ali ne? Uporabnikova odločitev ima najmanj dva možna negativna scenarija:

a) uporabnik dvomljivo vsebino kljub pomislekom odpre in v primeru škodljive vsebine tvega okužbo/nestabilnost sistema,
b) uporabnik vsebine ne odpre, a se ta izkaže za legitimno, zato lahko doživi poslovno škodo/izgubo.

1.8 Naša raziskovalna teza

Vsak uporabnik elektronske pošte bi moral imeti možnost varnega *predogleda* datotek PDF, preden jih prenese in ogled požene na svoji napravi. Še več, ta možnost bi morala obstajati neodvisno od starosti in vrste operacijskega sistema ali naprave, na kateri spremlja elektronsko pošto.

Hipoteza: neodvisno od uporabnikovega operacijskega sistema ali naprave je mogoče zasnovati sistem, ki z uporabo minimalnega `seccomp` profila, vsebnika Docker in principov uporabne varnosti zagotovi predogled PDF-ja brez tveganja za integriteto uporabnikove naprave – vse to v nekaj sekundah.

Predogled (tj. ogled brez interaktivne vsebine) zadošča, da se uporabnik seznani z vsebino in oceni verodostojnost dokumenta.

1.8.1 Rokovanje s sistemom: Dodatno uporabnost je mogoče doseči, če sistem teče v lokalnem oblaku. Standard mednarodne organizacije ISO 9241-11 rešuje težave, na katere uporabniki naletijo ob nepotrebnih korakih, ki so potrebni za dokončanje opravka [5]. V skladu z idejami standarda predlagamo naslednjo organizacijo sistema: organizacija vzpostavi e-naslov, na katerega član organizacije (npr. `janez.novak@x.org`) lahko posreduje sporočila s sumljivo priponko (npr. `predogled@x.org`). Za branje predala e-pošte naj v organizaciji skrbi poseben računalniški agent, ki pošto sprejme, jo varno odpre, pretvori morebitne priponke formata PDF v neizvršljivo datoteko formata BMP, JPG ali PNG po metodi, opisani v poglavju 2.4 in namesti pretvorjene datoteke na organizaciji dostopen strežnik HTTPS. Agentni sistem po pretvorbi odgovori na sporočilo in ob uspešni pretvorbi datotek, posreduje podatke za dostop do varnega ogleda datotek na strani strežnika HTTPS.

S tem se rokovanje s sistemomza uporabnika poenostavi na eno operacijo – posredovanje e-sporočila na zaupanja vreden naslov.

1.9 Prispevki članka

- Implementacija statičnega `pdftoppm` s `seccomp` v neprivilegiranem zabojniku Docker brez operacijskega sistema.
- Zasnova *forward-to-inspect* e-poštnega toka, ki sumljive PDF-je samodejno pretvori v JPG.
- Elaborirana primerjava s klasičnim VM-scenarijem (čas zagona, ocena kognitivne obremenitve, zanesljivost zadrževanja, vzdrževanje).

*Najpogosteje Windows ali namizno distribucijo Linuxa z grafičnim okoljem.

Organizacija članka: Metode so predstavljene v 2; diskusija z rezultati v 3; zaključek v poglavju 4.

2 METODE

V tem poglavju najprej natančno opisujemo predlagano metodo za varen predogled datoteke PDF, pozneje pa predlagano še metodo za rokovanje s sistemom.

2.1 Pretvorba datoteke PDF v neizvršljiv slikovni format

Osnovna ideja varnega predogleda datoteke PDF temelji na predpostavki, da je mogoče vsebino datoteke PDF pretvoriti v datotečni format, ki uporabnika varuje pred potencialno škodljivimi akcijami, omogoča pa neinteraktiven ogled vsebine. Če povprečen dokument PDF npr. pretvorimo v format slikovne datoteke (npr. nestisnjen format BMP/PPM ali stisnjen format JPG/PNG), si je vsebino mogoče ogledati s pomočjo programov za ogled fotografij. Po pretvorbi v slikovni format se izgubijo skriti elementi in izvršljiva koda, ki bi lahko prožili neavtorizirane akcije na uporabnikovem sistemu, saj slikovni formati ne podpirajo izvajanja naprednih akcij, kot je izvajanje kode JavaScript ali nalaganje z oddaljenega strežnika, manj verjetno je tudi, da bi uporabnik *kliknil na ponujeno hiperpovezavo* znotraj dokumenta. Za varen predogled vsebine je treba zagotoviti, da se proces pretvorbe PDF v slikovni format izvede po varnem postopku, katerega izvajanje samo po sebi ne more povzročiti škode uporabniku.

Za pretvarjanje datotek PDF v slikovne formate lahko uporabimo odprtokodni program `pdftoppm`, ki je del širše knjižnice `Poppler` za delo z datotekami PDF v sistemu Linux [33]. Je program, ki se zažene s pomočjo komandne vrstice, kot vhod sprejme datoteko formata PDF in jo sprva pretvori v format PPM (Portable Pixmap), ki ni stisnjen, naposled pa jo izda v želenem stisnjenem formatu (npr. JPG).

Odprtokodni program `pdftoppm` za svoje delo potrebuje kopico sistemskih knjižnic, ki jih v operacijski sistem Linux nameščamo prek t. i. repozitorijev. Vsaka od njih ima lahko svoje varnostne pomanjkljivosti, ki jih brez podrobne analize ne moremo poznati; še več, vsak trenutek se lahko koda v repozitoriju spremeni, po nadgradnji sistema pa naš proces lahko izrablja nov postopek, ki ga nismo varnostno preverili. Zato je treba odprtokodni program, ki ga želimo uporabljati za varnostno tvegane operacije, "zapakirati" v enotno izvršljivo datoteko, ki ji naposled ob izvajanju strogo omejimo dovoljene sistemske vire. Statično prevajanje takega tipa odprtokodnega programa zahteva svoje poglavje, ki sledi.

2.2 Postopek izdelave varnostno preverljive statične izvedbe `pdftoppm`

Kompilacija knjižnic PDF-procesorjev, kakršen je `pdftoppm` iz projekta *Poppler*, na klasičnih distribucijah GNU/Linux pogosto propade zaradi kompleksnih odvisnosti in verzijskih neskladij dinamičnih knjižnic. V varnostnih okoljih, kjer si prizadevamo za deterministične, reproducibilne in zanesljive gradnje, so takšne neuspešne kompilacije nesprejemljive, saj otežujejo naknadno analizo, preverjanje ranljivosti in zanesljivo posodabljanje.

Da bi zagotovili izvajanje orodja `pdftoppm` brez vsakršnih sistemskih odvisnosti, smo uporabili pristop *Docker večstopenjske gradnje*. V prvi stopnji (oznaka `builder`) prevedemo in povežemo vse potrebne knjižnice in program sam, v drugi stopnji (`scratch`) pa v končno podobo vključimo zgolj en artefakt:

```
########## builder ##########
FROM alpine:edge AS builder
ENV  MAKEFLAGS="-j$(nproc)"          \
     PKG_CONFIG="pkg-config --static"\
     CFLAGS="-static -O2 -pipe"      \
     CXXFLAGS="${CFLAGS}"            \
     LDFLAGS="-static -s"

RUN apk add --no-cache --virtual .build-deps \
        build-base git cmake ninja meson      \
        libtool autoconf automake pkgconf     \
        bison flex wget xz tar gzip
```

Nastavitev okolja (`CFLAGS=-static`) prisili *statično povezovanje* vseh artefaktov; parameter `-s` hkrati odstrani simbole za razhroščevanje, kar zmanjša površino za obratni inženiring.

2.2.0-A) Kompilacija vseh odvisnosti v statični obliki: Zaporedoma prevajamo in namestimo `zlib`, `libjpeg-turbo`, `libpng`, `Brotli`, `FreeType` ter končno `Poppler`. Za vsako knjižnico izrecno onemogočimo deljene (*shared*) objekte in izberemo statično povezovanje (`--enable-static`, `ENABLE_SHARED=FALSE`, ...). Ključni parametri pri konfiguraciji `Poppler` so:

- `-DBUILD_SHARED_LIBS=OFF`: generira zgolj statične arhive;
- `-DENABLE_...=OFF`: izključimo nepotrebne module (QT, GLib, Cairo ipd.), s čimer zmanjšamo napadno površino;
- `-DCMAKE_EXE_LINKER_FLAGS= -static -lbrotlidec -lbrotlicommon`: ročno dodamo statični dekoder Brotli, ki ga `FreeType` potrebuje pri obdelavi kompresiranih pisav.

2.2.0-B) Minimizacija artefakta: Po uspešni kompilaciji odstranimo vse preostale deljene knjižnice in razvojna orodja:

```
find /usr/local/lib -name '*.so*' -delete
apk del .build-deps
```

S tem zagotovimo, da vdelana podoba ne vsebuje odvečnih binarnih datotek, ki bi lahko pomenile varnostna tveganja (npr. zaradi slabše vzdrževanih skupnih knjižnic).

2.2.0-C) Končna podoba.: Na drugi stopnji uporabimo bazo vsebnika `scratch`, ki je povsem prazna:

```
########## final ##########
FROM scratch
COPY --from=builder /usr/local/bin/pdftoppm /pdftoppm
ENTRYPOINT ["/pdftoppm", "-h"]
```

Tako nastala kontejnerska slika vsebuje *natanko eno* statično povezano izvršljivo datoteko, kar bistveno poenostavi preverjanje integritete in izključuje napade, ki temeljijo na nadomeščanju deljenih knjižnic.

2.2.0-D) Operativna uporaba: Kontejner lahko zaženemo neposredno:

```
docker run --rm -v "$(pwd)":/data poppler-static \
           /pdftoppm /data/vhod.pdf /data/izhod
```

Zaradi statičnega povezovanja je mogoče binarno datoteko varno izvleči in pognati tudi zunaj kontejnerja na poljubnem sodobnem sistemu x86_64 Linux:

```
cid=$(docker create poppler-static)
docker cp "$cid:/pdftoppm" ./pdftoppm \
  && docker rm "$cid"
./pdftoppm -v   # deluje brez odvisnosti
```

2.2.0-E) Varnostni pomen: Izbrani pristop prinaša sledeče prednosti v kontekstu računalniške varnosti:

a) **Determinističnost.** Docker reproducira natanko isto okolje gradnje, kar omogoča zanesljivo revizijo in preverjanje ranljivosti (npr. CVE-2023-41360 v `libpng` [6]).
b) **Minimalna napadna površina.** Slika `scratch` + statični binarij izključuje nepotrebne interprete, paketne menedžerje in dinamične knjižnice.
c) **Odpornost proti napakam povezanih knjižnic.** Vse odvisnosti so eksplicitno fiksirane, kar onemogoča podtikanje zlonamernih paketov.
d) **Lažja forenzična analiza.** Eno samo binarno datoteko je enostavneje podvreči statični analizi (npr. `gdb`, `ghidra`), saj ni potrebe po dinamičnem nalaganju dodatnih modulov.

S tem postopkom smo torej izpolnili robustne varnostne zahteve in zagotovili, da je orodje `pdftoppm` varno izvedljivo v okoljih, kjer so prisotne stroge omejitve glede nameščanja zunanjih knjižnic (npr. v *sandboxih*, pri obdelavi potencialno zlonamernih PDF dokumentov ali na visoko heterogenih produkcijskih sistemih). Velikost prevedenega in statično povezanega orodja `pdftoppm` znaša zgolj 38,8 MB.

2.3 Implementacija procesnih omejitev

Neposreden zagon statično prevedenega orodja `pdftoppm` še ne zagotavlja popolne varnosti matičnega sistema. Med izvajanjem programa je treba dodatno omejiti vse procese, ki bi potencialno lahko škodili sistemu in jih za osnoven predogled vsebine ne potrebujemo. Za zagotovitev varnostnega okolja je nujno vzpostaviti mehanizme, ki *zaklenejo* (angl. *lockdown*) nedovoljene sistemske klice in omejijo dostop do sistemskih virov. V operacijskem sistemu Linux to dosežemo z integracijo dveh ključnih mehanizmov:

2.3.1 Omejevanje sistemskih klicev s `seccomp`: Knjižnica `seccomp` (Secure Computing Mode) [12] je jedrni mehanizem za prehod procesa v varno stanje, kjer so privzeto onemogočeni vsi sistemski klici razen osnovnih operacij (npr. sporočila o stanju in izhodu procesa ter osnovni operaciji I/O). Pri načrtovanju pravilnika za `seccomp` moramo upoštevati funkcionalne zahteve ciljne aplikacije, v našem primeru procesa `pdftoppm`. Naslednji primer demonstrira konfiguracijo, ki:

- Dovoli operaciji odpiranja (`open`) in zapiranja (`close`) datotek
- Strogo prepove kloniranje procesov (`vfork`) in ustvarjanje map (`mkdir`)

```
#include <seccomp.h>
scmp_filter_ctx const ctx =
              seccomp_init(SCMP_ACT_TRAP);
seccomp_rule_add(ctx,SCMP_SYS(open),
                     SCMP_ACT_ALLOW,0);
seccomp_rule_add(ctx,SCMP_SYS(close),
                     SCMP_ACT_ALLOW,0);
seccomp_rule_add(ctx,SCMP_SYS(vfork),
                     SCMP_ACT_ERRNO(0),0);
seccomp_rule_add(ctx,SCMP_SYS(mkdir),
                     SCMP_ACT_ERRNO(0),0);
```

2.3.2 Nadzor sistemskih virov s `setrlimit`: Mehanizem `setrlimit(2)` omogoča granularno omejevanje porabe računalniških virov. Omejitve procesorskega časa (`RLIMIT_CPU`) se upravljajo z dvojno mejo:

- *Mehka meja* (2 sekundi): Ob presežku proces prejme signal `SIGXCPU`.
- *Trda meja* (3 sekunde): Dokončno prekinitev procesa s signalom `SIGKILL`.

```
#include <sys/time.h>
#include <sys/resource.h>
setrlimit(RLIMIT_CPU, 2, 3);
```

Podobno lahko vzpostavimo omejitve za:

- Velikost ustvarjenih datotek (`RLIMIT_FSIZE`)
- Pomnilniški prostor za podatkovni segment (`RLIMIT_DATA`)

```
int slim = 10*1024*1024;
int hlim = 12*1024*1024;
setrlimit(RLIMIT_FSIZE, slim, hlim);
setrlimit(RLIMIT_DATA, slim, hlim)
```

V praksi je treba omejitve nastaviti tako, da ne bodo blokirale pretvorbe legitimnih datotek PDF večjih velikosti.

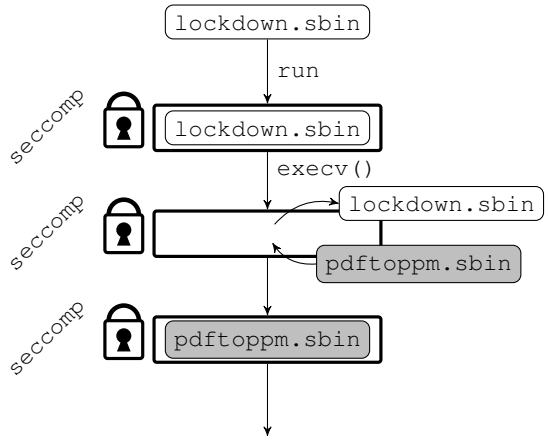
2.3.3 Zagon nezaupanja vrednega procesa: Po vzpostavitvi varnostnih mehanizmov izvedemo nezaupanja vreden proces z uporabo sistemskega klica `execv`. Ta ohrani obstoječi identifikator procesa (PID) in varnostne omejitve, zamenja pa izvajano kodo:

```
execv(argv[1], argv + 1);
```

kjer `argv[1]` predstavlja pot do binarne izvršljive datoteke, `argv + 1` pa kazalec na njen argumentni

niz, v katerem so navedene nastavitve in poti vhodnih ter izhodnih datotek. Ta pristop zagotavlja, da se vse predhodno konfigurirane omejitve nanašajo na ciljni proces.

Kodo, ki vsebuje pravila za `seccomp` `lockdown.c` prevedemo in jo statično povežemo s prevedeno matično knjižnico, tako da dobimo eno izvršljivo datoteko `lockdown.sbin` (0,87 MB). Zagon `lockdown.sbin` sproži zagon programa, podanega preko parametra (`pdftoppm.sbin`) pod omejenimi procesnimi pogoji (slika 2).



Slika 2 Zaklepanje procesa – program `lockdown.c` zaklene dovoljenja za proces, ukaz `execv()` nadomesti program, ki trenutno teče, z binarno statično povezano knjižnico `pdftoppm.sbin` pod isto (starševsko) procesno identiteto.

2.4 *Zagon v vsebniku in izvlečenje izhodnih datotek*

Naš predlog za zagon varnega sistema pretvorbe PDF v JPG predvideva zagon statično povezane knjižnice `pdftoppm.sbin` pod `seccomp.sbin` znotraj minimalnega vsebnika Docker. V tem poglavju opisujemo predlog načina uporabe vsebnika Docker za zagon predogleda datotek PDF. Predvidevamo, da že obstajata binarni datoteki `lockdown.sbin` ter `pdftoppm.sbin`. V sliko Docker ju namestimo z naslednjim ukazom za Docker:

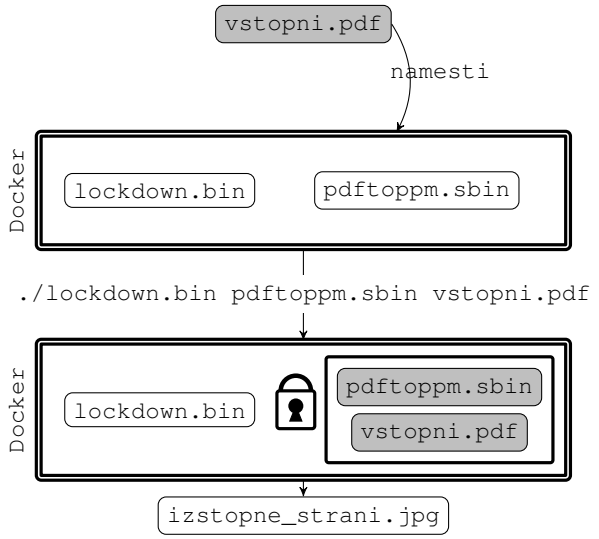
```
#seccomp-pdf/Dockerfile
FROM scratch

COPY lockdownsl /usr/local/bin/lockdownsl
COPY pdftoppm  /usr/local/bin/pdftoppm

WORKDIR /work

# zazenemo:  lockdownsl
/usr/local/bin/pdftoppm  <all user args>
ENTRYPOINT ["/usr/local/bin/lockdownsl", \
            "/usr/local/bin/pdftoppm"]
```

Tako dobimo sliko vsebnika brez nameščenega operacijskega sistema, brez knjižnic, brez interpreterja za `shell/bash`, skratka golo okolje z minimalno napadno površino.

Na matičnem sistemu ustvarimo ciljno mapo za izhodne datoteke, zaženemo vsebnik Docker, pripnemo vhodno datoteko in pripnemo izhodno mapo ter Dockerju posredujemo parametre, ki naj se zaženejo v zaklenjenem procesu:

```
mkdir -p out/
docker run --rm \
  -v "$PWD/in.pdf":/work/in.pdf:ro \
  -v "$PWD/out":/work/out          \
  pdftoppm-seccomp:latest          \
     -jpeg -jpegopt quality=50     \
     -r 300                        \
     /work/in.pdf     /work/out/page
```

Po zagonu zgornjih navodil za `bash` se ob uspešni pretvorbi vsaka stran vhodnega PDF pojavi v obliki datotek JPG v mapi `out`, kjer so pripravljene za varen predogled. Slika 3 predstavlja potek procesa zagona pretvorbe PDF v JPG znotraj vsebnika.



Slika 3 Zagon zaklenjenega procesa `pdftoppm` znotraj vsebnika.

2.5 *Zasnova toka `forward-to-inspect`*

Za namen maksimizacije varnosti za uporabnika ter hkrati dobrega načina za rokovanje s sistemom predlagamo, da pretvornik PDF v JPG deluje v sklopu izoliranega strežniškega sistema v oblaku matične organizacije. Nevarnost, ki preti iz zagona neznanih datotek (e-poštnih priponk), je usmerjena na uporabnika v organizaciji — zato je pomembno, da ergonomiko sistema osredinimo prav na uporabnika. V našem predlogu predvidevamo, da ima uporabnik svobodo prejemanja e-poštnih sporočil z različnih, tako zunanjih kot notranjih naslovov.

2.5.1 Organizacija oblaka organizacije: Oblačni strežniški sistem znotraj organizacije naj bo konfiguriran tako, da sprejema e-poštna sporočila s svojih (ali registriranih) domen. Računalniški agent naj prejeto sporočilo

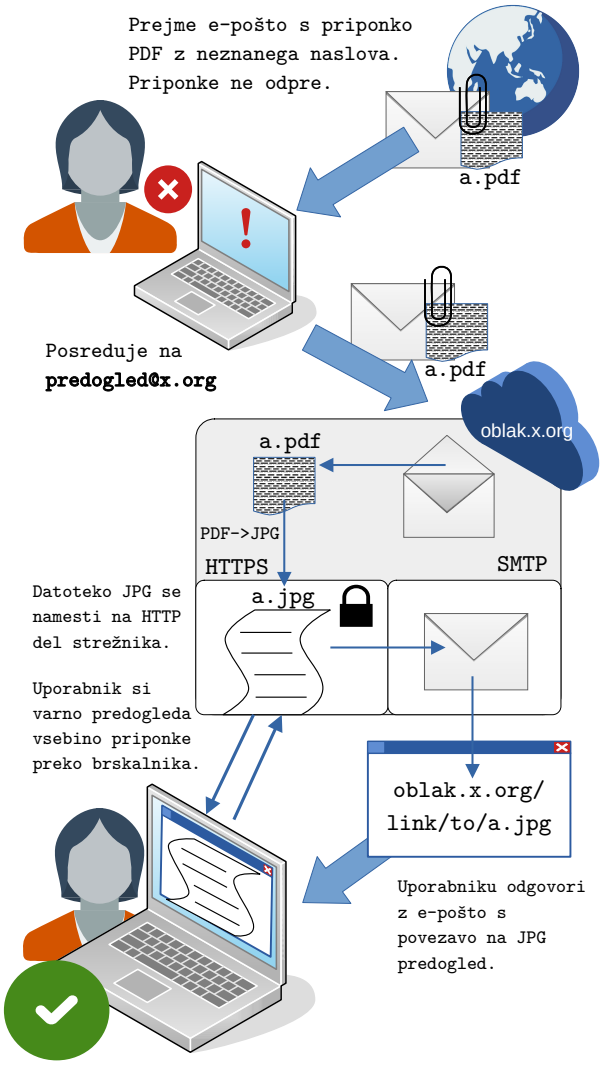
analizira. V primeru obstoječih priponk PDF naj izvede naslednji algoritem:

1.) Priponko `a.pdf` (ali priponke, če jih je več) naj namesti v začasno mapo z minimalnimi pravicami znotraj strežnika.
2.) Izvede naj varno pretvorbo PDF v JPG z uporabo metode, opisane v 2.4.
3.) Izhodne datoteke tipa JPG naj namesti na del strežnika dostopen prek HTTPS.
4.) Del strežnika, ki skrbi za elektronsko pošto (npr. SMTP), naj uporabniku odgovori z e-poštnim sporočilom, v katerem so navedeni hiperpovezava in morebitni ključi za dostop, ki vodijo do zasebne galerije za ogled vsebin oblike JPG (predogled prvotne datoteke PDF).

2.5.2 Uporabnikov vidik: Uporabnik e-pošte ob prejemu pošte neznanega pošiljatelja s sumljivo priponko PDF izvede naslednje korake:

1.) Celotno e-poštno sporočilo posreduje (»*forward*«) na poseben naslov znotraj organizacije (npr. `predogled@x.org`).
2.) Od oblačne storitve uporabnik prejme odgovor, ki vsebuje osnovne informacije o uspehu pretvorbe PDF v predogled JPG. Sporočilo vsebuje povezavo na strežnik iste organizacije, ki ga uporabnik odpre v brskalniku.
3.) Prek brskalnika si ogleda vse strani izvorne datoteke PDF, pretvorjene v neizvršljiv format JPG.
4.) Na podlagi verodostojnosti vsebine se lahko uporabnik odloči, ali bo izvorni dokument PDF naposled na svoji napravi zagnal ali pobrisal sporočilo ter, ne nazadnje, ali bo nadaljeval s korespondenco s pošiljateljem.

Celoten predlagani tok je orisan na sliki 4.



Slika 4 Zasnova toka *Forward-to-inspect*. Uporabnik prejme e-pošto s sumljivo priponko. Posreduje jo na naslov znotraj organizacije, ki je namenjen varnemu odpiranju priponk PDF. Oblačni sistem datoteko po predlagani varni metodi pretvori v neizvršljiv format (npr. JPG), ga namesti na del strežnika, ki ga lahko servira preko HTTPS. Uporabniku odgovori na e-poštno sporočilo in v njem posreduje povezave na varen predogled priponke, ki se ga opravi preko brskalnika.

3 Rezultati in diskusija

3.1 Časovna primerjava

V tem poglavju kvantitativno prikažemo hitrost pretvorbe datotek PDF v JPG z uporabo izoliranega peskovnika s statično zgrajeno knjižnico `pdftoppm` s `seccomp`. Opravili smo tri eksperimente:

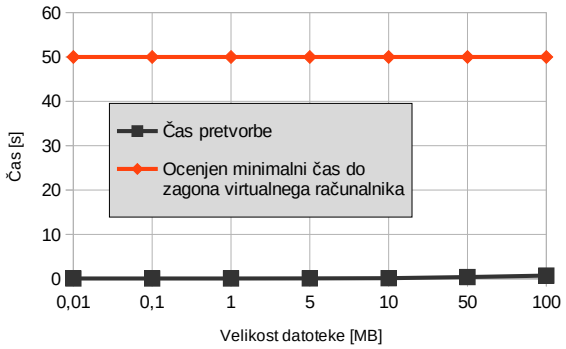
1.) testiranje pretvorbe z enostranskimi datotekami PDF različnih velikosti datoteke,
2.) test z večstranskimi datotekami PDF različnih velikosti datoteke ter
3.) test z izrisovanjem transparentnih večslojnih datotek PDF (procesno zahteven del je mešanje barve za vsako piko izrisa).

Za zagon eksperimentov smo vsakokrat uporabili statično prevedeno orodje `pdftoppm` v `seccomp` peskovniku. Vsakokrat smo pretvorbene parametre nastavili na 50% kompresiranje JPG ter renderiranje z 80 pikami na palec (80 ppi). Pretvorba se je zagnala na operacijskem sistemu Linux Debian, na prenosnem računalniku s procesorjem razreda Intel i5.

Ugotovili smo naslednje: Eksperiment št. 1 je pokazal, da se enostranske datoteke velikosti od 0,01 do 100 MB v datoteko JPG pod zaklenjenimi procesnimi pogoji pretvorijo v manj kot sekundi (tabela 1 ter slika 5). Večstranske datoteke (eksperiment št. 2) za pretvorbo zahtevajo več časa – če ima datoteka PDF 50 strani in je velika 5 MB, je čas pretvorbe 2,57 sekunde (tabela 2 ter slika 6). Najbolj zahteven test v tej seriji je bil pretvorba datoteke 100 MB s 1.000 stranmi – tu je peskovnik `seccomp` ustavil pretvorbo ob prekoračitvi 30-sekundnega pravila izvajanja procesa. Kljub dolgemu izvajanju se je uspešno pretvorilo 563 strani, kar upo-

Velikost izv. datoteke [MB]	Čas pretvorbe [s]
0,01	0,06
0,1	0,06
1	0,06
5	0,09
10	0,13
50	0,38
100	0,72

Tabela 1 Tabela skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku - test z enostranskimi datotekami PDF različnih velikosti datoteke.



Slika 5 Graf skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku v primerjavi z ocenjenim minimalnim časom do zagona virtualnega računalnika - test z enostranskimi datotekami PDF različnih velikosti datoteke. Izvajanje pretvorbe datoteke velikosti 100 MB (ena stran) v preizkušanju traja 0,72 s.

rabnik kljub vsemu lahko uporabi za grobi predogled datoteke. Zadnji eksperiment v tej seriji je preizkušal trajanje izrisovanja velikega števila slojev enostranskega dokumenta PDF. Izrisovanje strani z 10 sloji je trajalo 0,48 sekund, medtem ko je zahtevnejši primer s 160 sloji zahteval 7,72 sekunde za pripravo predogleda.

Z uporabo predlagane metode za pretvorbo PDF v JPG (2.4) imamo predogled vsebine torej opravljen v razredu sekund, tipični tekstovni dokumenti z eno do dvema stranema se v JPG pretvorijo v času, krajšem od sekunde, na računalniku s procesorjem razreda Intel i5. Zaradi kratkega časa izvajanja bi lahko predlagano metodo pretvorbe dokumenta v peskovniku uporabniki poganjali tudi na lokalnem računalniku (znotraj operacijskega sistema Linux) kot vtičnik v odjemalniku elektronske pošte ali kot samostojno aplikacijo.

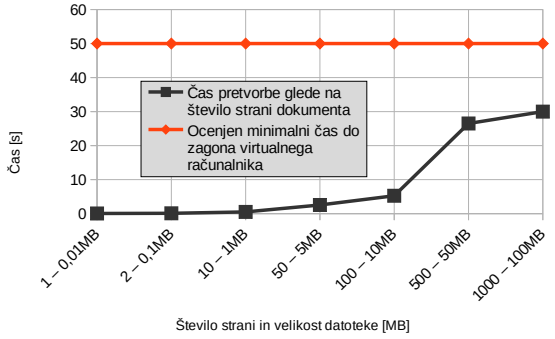
Klasična metoda z uporabo virtualnega računalnika (VM) traja **najmanj 50 sekund** od zagona sistema do predogleda datoteke PDF na Intelu i5, z gostom Linux Debian z lahkim Xfce grafičnim okoljem (optimalen scenarij glede hitrosti). Metodo je mogoče avtomatizirati prek uporabe ukazov npr. `VboxManage`, vendar njihova uporaba ne prinese signifikantnih časovnih prihrankov, saj postopek večino časa namreč porabi za zagon celotnega operacijskega sistema [10], [18]. Po uporabi virtualnega računalnika je treba še ponastaviti posnetek (*snapshot*) sistema na začetno stanje, kar pomeni dodatnih 30 sekund ročnega dela in prinaša možnost za napake človeškega faktorja (zagon napačnega posnetka, opustitev ponastavljanja posnetkov itd.).

3.2 Kognitivna obremenitev

Scenarij z uporabo virtualnega računalnika za predogled potencialno nevarnih datotek zahteva strokovnjaka in veliko natančnosti za varno upravljanje datotek. Kognitivno breme za predogled datotek na ta način je lahko preveliko za tipičnega uporabnika elektronske pošte. Če ima uporabnik na voljo oblačni sistem z vmesnikom tipa *forward-to-inspect*, kot predlagano v poglavju 2.5, je celoten kognitivni napor uporabnika zmanjšan na en klik na tipko *posreduj* (sumljiva datoteka v priponki se posreduje na poseben naslov agenta elektronske pošte) ter klik na povezavo za ogled statično pretvorjenih dokumentov v povratnem sporočilu. Ocenjujemo, da bi

Velikost izv. datoteke [MB]	Število strani	Čas pretvorbe [s]
0,01	1	0,05
0,1	2	0,1
1	10	0,5
5	50	2,54
10	100	5,25
50	500	26,5
100	1000	30

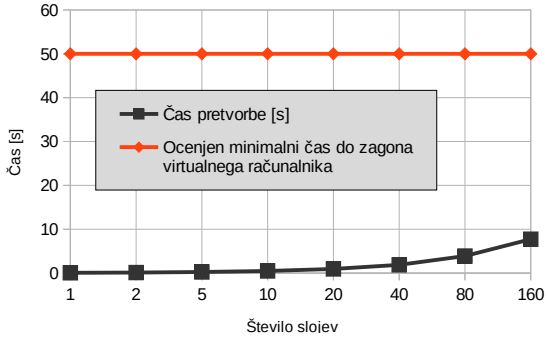
Tabela 2 Tabela skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku - test z večstranskimi datotekami PDF različnih velikosti datoteke. V zadnjem primeru (1000 strani, 100 MB) je prišlo do ustavitve procesa pri vnaprej postavljeni časovni omejitvi izvajanja peskovnika 30 s. Do takrat je sistem skupno pretvoril 563 strani, kar še vedno zadostuje za predogled verodostojnosti vsebine.



Slika 6 Graf skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku v primerjavi z ocenjenim minimalnim časom do zagona virtualnega računalnika - test z večstranskimi datotekami PDF različnih velikosti datoteke. V zadnjem primeru (1000 strani, 100 MB) je prišlo do ustavitve procesa pri vnaprej postavljeni časovni omejitvi izvajanja peskovnika 30 s.

Število slojev	Čas pretvorbe [s]
1	0,06
2	0,11
5	0,25
10	0,48
20	0,94
40	1,87
80	3,88
160	7,72

Tabela 3 Tabela skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku - test z večslojnimi datotekami PDF.



Slika 7 Graf skupnega časa izvajanja pretvorbe PDF v JPG v zaklenjenem peskovniku v primerjavi z ocenjenim minimalnim časom do zagona virtualnega računalnika - test z večslojnimi datotekami PDF.

tovrstni sistem lahko uporabil vsak povprečen uporabnik osebnega računalnika. Predlagana metoda z izvajanjem v oblaku se lahko uporabi tako na osebnih računalnikih kot tablicah in mobilnih telefonih, *neodvisno od operacijskega sistema*, za kar verjamemo, da je najmočnejša točka predlaganega pristopa.

3.3 Zanesljivost

Zanesljivost zadrževanja grožnje ocenjujemo z vidika napadalne površine, ki je nezaupanja vrednemu procesu na voljo. Z uporabo predlagane metode pretvorbe z uporabo Docker, zaklenjenega peskovnika seccomp in statično povezanih knjižnic je napadalna površina minimizirana v največji meri, saj vsebuje le knjižnice, ki so nujno potrebne za pretvorbo iz PDF v JPG. Dovolijo se le procesi, ki so smiselni za izvedbo naloge pretvorbe; vsi preostali procesi so blokirani. V primeru uporabe polne virtualizacije je procesu ogleda PDF na voljo celoten operacijski sistem, kar bi lahko zlonamerni kodi iz dokumenta PDF omogočilo večje možnosti za uspeh. Našo metodo smo preizkušali z nabori javno dostopnih datotek PDF, namenjenih za penetracijsko testiranje [8], [29], [1]. Nobena od preizkušenih zlonamernih datotek PDF ni vplivala na delovanje operacijskega sistema.

3.4 Reprodukcija originala

Format PDF predstavlja širok nabor funkcij – nekatere se s pretvorbo v format JPG izgubijo. Prav tako se izgubijo določene funkcionalnosti, ki se zanašajo na celovitost zalednega operacijskega sistema (souporaba sistemskih pisav, dostop do lokalnih datotek, dostop do spleta ...). Če imajo te funkcionalnosti v dokumentu PDF miroljubni namen, jih s pretvorbo po naši metodi izgubimo in si dokumenta ne moremo verodostojno ogledati. To je lahko ena od pomanjkljivosti, na katero lahko naletijo uporabniki predlagane metode – zagotovo pa so mogoče tudi izboljšave v tej smeri. Med testiranjem je imelo nekaj sicer neškodljivih datotek popačen nabor pisav.

V tabeli 4 kvalitativno primerjamo lastnosti predlagane metode namenskega peskovnika z metodo uporabe virtualnega računalnika za predogled nezaupanja vrednih datotek PDF.

4 Zaključek

Odpiranje datotek PDF iz neznanih virov je tvegano. V pričujočem prispevku smo predstavili metodo, ki ponuja varen predogled datotek PDF tako, da zagotovi varno okolje za njihovo pretvorbo v format JPG. S tem se odpravi možnost uspešnega izvajanja zlonamerne kode znotraj ogledovalnikov PDF. Če sistem organiziramo v obliki *forward-to-inspect*, kjer se nezaupanja vredne priponke PDF posredujejo v oblačni sistem za prevedbo v JPG, takšna rešitev sledi načelom uporabne varnosti, saj minimalizira napadalno površino brez ogrožanja uporabniške izkušnje ali potrebe po dodatnem znanju uporabnika.

Primerjava s klasičnim VM-scenarijem je pokazala, da predlagani pristop dosega boljše rezultate z vidika odzivnosti, kognitivne enostavnosti in stroškov vzdrževanja, hkrati pa zagotavlja enakovredno ali višjo stopnjo zanesljivosti pri obravnavi potencialno nevarnih dokumentov.

Zlasti pomembno je, da metoda ne predpostavlja zaupanja obravnavanemu dokumentu, temveč v celoti temelji na pasivni vizualizaciji vsebine. To ustreza sodobnim načelom ničelnega zaupanja (*zero-trust*) ter zagotavlja ločitev med vsebino in procesnim okoljem.

Gre za namensko optimiziran pristop, primeren za okolja, kjer so pomembne zanesljivost, sledljivost in minimalna kompleksnost – na primer v visokošolskem, raziskovalnem ali industrijskem kontekstu. Metodo je mogoče aplicirati na uporabnike vseh elektronskih komunikacijskih naprav: osebnih računalnikov, tabličnih računalnikov ali pametnih mobilnikov, bodisi kot razširitev ogledovalnikov elektronske pošte ali kot samostojno oblačno storitev.

V prihodnje bi bilo smiselno vključiti podporo za optično prepoznavanje besedila in raziskati avtomatizirano integracijo z obstoječimi učnimi ali dokumentnimi sistemi ter analizirati obnašanje v primeru kompleksnih PDF-struktur.

Prepričani smo, da bo naš predlog industrijskim partnerjem in raziskovalcem pomagal pri razvoju nadaljnjih

Tabela 4 Kvalitativna primerjava predlagane metode in klasičnega VM-scenarija

Kriterij	**Predlagana metoda (PDF → JPG predogled)**	**Klasični VM-scenarij**
Čas izvedbe	Predogled v sekundah ali manj (krajši dokumenti se pretvorijo prej)	Zagon in prikaz VM traja najmanj več deset sekund ali nekaj minut (tudi za majhne datoteke PDF)
Kognitivna obremenitev	Nizka – enostaven vmesnik, brez dodatne konfiguracije	Višja – potrebna orientacija v okolju VM, več korakov, potrebna pravilna konfiguracija deljene mape, zaprt internet, ipd.
Velikost napadalne površine	Minimalna - proces teče v zaklenjenem okolju, na voljo so le nujno potrebne knjižnice	Srednja – napadalec lahko izkoristi celovitost sistema VM
Vzdrževanje	Minimalno – lokalna knjižnica, brez odvisnosti od verzij operacijskega sistema	Zahtevno – redna posodobitev operacijskega sistema
Reprodukcija vsebine	Zmanjšana	Popolna

rešitev za doseganje robustnosti in odpornosti proti trenutnim izzivom informacijske varnosti.

DOSTOPNOST RAZISKOVALNIH PODATKOV

Programska koda in podatki, ki omogočajo reprodukcijo zaključkov iz pričujočega dela so dostopni v naslednjih arhivih: https://github.com/zigarojec/staticpdftoppm-EV2025 (statično prevajanje knjižnice Poppler ter Seccomp), https://doi.org/10.5281/zenodo.17629417 (PDF datoteke za testiranje časovne zahtevnosti).

LITERATURA

[1] PDF Insecurity Website, June 2021. https://pdf-insecurity.org/downloads/paper_reports_theses.html.

[2] Acrobat Developer Resources Acrobat Developer Docs, December 2024. https://opensource.adobe.com/dc-acrobat-sdk-docs.

[3] Desktop Windows Version Market Share Worldwide | Statcounter Global Stats, June 2025. [Online; accessed 10. Jun. 2025].

[4] Document management — Portable document format, June 2025. https://www.iso.org/standard/51502.html.

[5] Ergonomics of human-system interaction, part 110: Interaction principles, June 2025. https://www.iso.org/standard/77520.html.

[6] NVD - CVE-2023-41360, July 2025. https://nvd.nist.gov/vuln/detail/CVE-2023-41360.

[7] PDF in 2016: Broader, deeper, richer – PDF Association, May 2025. https://pdfa.org/pdf-in-2016-broader-deeper-richer/.

[8] 0ca. corpus_pdfs, July 2025. https://github.com/0ca/corpus_pdfs?tab=readme-ov-file.

[9] ANY.RUN. Any.run – interactive online malware analysis sandbox, 2025. Accessed 10 Jun 2025.

[10] Ridlo Sayyidina Auliya, Yen-Lin Lee, Chia-Ching Chen, Deron Liang, and Wei-Jen Wang. Analysis and prediction of virtual machine boot time on virtualized computing environments. *J. Cloud Comp.*, 13(1):1–21, December 2024.

[11] Nguyen Tan Cam, Tran Quang Hung, and Pham Tien Nam. uitPDF-MalDe: Malicious Portable Document Format files detection using multi machine learning models. *Eng. Appl. Artif. Intell.*, 143:110031, March 2025. Publisher: Pergamon.

[12] Claudio Canella, Mario Werner, Daniel Gruss, and Michael Schwarz. Automating Seccomp Filter Generation for Linux Applications. *arXiv preprint arXiv:2012.02554*, 2020.

[13] rohann@checkpoint com. The Weaponization of PDFs : 68% of Cyber attacks begin in your inbox, with 22% of these hiding in PDFs. *Check Point Blog*, April 2025. https://blog.checkpoint.com/research/the-weaponization-of-pdfs-68-of-cyberattacks-begin-in-your-inbox-with-22-of-these-hiding-in-pdfs.

[14] CrowdStrike Inc. Hybrid analysis – free falcon sandbox malware analysis service, 2025. Accessed 10 Jun 2025.

[15] K. D. E. e.V. Okular, Version 24.12, 2024. https://okular.kde.org/.

[16] Di Feng, Min Yu, Yongjian Wang, Chao Liu, and Chunguang Ma. Detecting Malicious PDF Files Using Semi-Supervised Learning Method. In B Wu and QM Lu, editors, *5TH INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER SCIENCE APPLICATIONS AND TECHNOLOGIES (ACSAT 2017)*, pages 1–9, 2017.

[17] FileScan GmbH. Filescan.io – next-generation malware analysis platform, 2025. Accessed 10 Jun 2025.

[18] Yatheendraprakash Govindaraju, Hector Duran-Limon, and Efrén Mezura-Montes. A regression tree predictive model for virtual machine startup time in IaaS clouds. *Cluster Computing*, 24:1–17, June 2021.

[19] Norman A. Graf. Extra Dimensions: 3D in PDF Documentation. In *INTERNATIONAL CONFERENCE ON COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS 2012 (CHEP2012), PTS 1-6*, volume 396 of *Journal of Physics Conference Series*. Brookhaven Natl Lab (BNL); ACEOLE; Data Direct Networks; Dell; European Middleware Initiat; Nexsan, 2012. ISSN: 1742-6588.

[20] Leonid Grustniy. Top 4 dangerous file attachments. *Kaspersky*, November 2019. https://www.kaspersky.com/blog/top4-dangerous-attachments-2019/27147.

[21] Adobe Inc. Adobe Acrobat Reader DC, Version 25.001.20521, 2025. https://helpx.adobe.com/acrobat/release-note/release-notes-acrobat-reader.html.

[22] Apryse Software Inc. Xodo PDF Reader & Editor (Android app), 2025. https://play.google.com/store/apps/details?id=com.xodo.pdf.reader.

[23] Foxit Software Inc. Foxit PDF Reader, Version 2025.1.0.27937, 2025. https://www.foxit.com/pdf-reader/version-history.html.

[24] Tommaso Innocenti, Louis Jannett, Christian Mainka, Vladislav Mladenov, and Engin Kirda. "Only as Strong as the Weakest Link": On the Security of Brokered Single Sign-On on the Web. In *IEEE Symposium on Security and Privacy (S&P)*, pages 24–24. IEEE Computer Society, May. https://www.computer.org/csdl/proceedings-article/sp/2025/223600a024/21B7Qfd8kiA.

[25] Joe Security LLC. Joe sandbox cloud basic – automated malware analysis, 2025. Accessed 10 Jun 2025.

[26] Krzysztof Kowalczyk. SumatraPDF, Version 3.5.2, 2023. https://www.sumatrapdfreader.org/download-free-pdf-viewer.html.

[27] Daiping Liu, Haining Wang, and Angelos Stavrou. Detecting Malicious Javascript in PDF through Document Instrumentation. In *2014 44TH ANNUAL IEEE/IFIP INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS (DSN)*, International Conference on Dependable Systems and Networks, pages 100–111. IEEE; IFIP; IEEE Comp Soc, 2014. ISSN: 1530-0889.

[28] Artifex Software LLC. MuPDF viewer (Android app), 2025. https://play.google.com/store/apps/details?id=com.artifex.mupdf.viewer.app.

[29] luigigubello. PayloadsAllThePDFs, July 2025. https://github.com/luigigubello/PayloadsAllThePDFs.

[30] Christian Mainka, Vladislav Mladenov, and Simon Rohlmann. Shadow Attacks: Hiding and Replacing Content in Signed PDFs. In *28TH ANNUAL NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM (NDSS 2021)*, 2021.

[31] Microsoft. Anti-malware protection in exchange online and exchange online protection, 2025. Describes EOP malware scanning pipeline, multiple engines, quarantine, and policy options.

[32] Jens Müller, Dominik Noß, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Processing Dangerous Paths - On Security and Privacy of the Portable Document Format. In *Network and Distributed System Security Symposium*, Virtual Conference, February 2021. Internet Society. https://www.ndss-symposium.org/ndss-paper/processing-dangerous-paths-on-security-and-privacy-of-the-portable-document-format/.

[33] Poppler Developers. pdftoppm: PDF to Image Conversion Utility, 2025. https://linuxcommandlibrary.com/man/pdftoppm.

[34] GNOME Project. Evince, Version 48.0, 2025. https://launchpad.net/ubuntu/+source/evince/+changelog.

[35] Karen Scarfone, Murugiah Souppaya, and Paul Hoffman. Guide to Security for Full Virtualization Technologies. *CSRC \vert NIST*, January 2011.

[36] Adobe Systems. Fast facts, June 2025. https://www.adobe.com/content/dam/cc/en/fast-facts/pdfs/fast-facts.pdf.

[37] Mesut Togacar and Burhan Ergen. Processing 2D barcode data with metaheuristic based CNN models and detection of malicious PDF files. *APPLIED SOFT COMPUTING*, 161, August 2024.

[38] Jose Torres and Sergio De Los Santos. Malicious PDF Documents Detection using Machine Learning Techniques A Practical Approach with Cloud Computing Applications. In P Mori, S Furnell, and O Camp, editors, *ICISSP: PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS SECURITY AND PRIVACY*, pages 337–344, 2018.

[39] VirusTotal (Google LLC). Virustotal private scanning, 2025. Accessed 10 Jun 2025.

[40] Google Workspace. How gmail helps protect against viruses and malware, 2025. Describes Gmail's automatic scanning of attachments and handling of infected files.

Žiga Rojec je leta 2014 diplomiral, leta 2018 pa doktoriral na Fakulteti za elektrotehniko Univerze v Ljubljani. Od leta 2014 je zaposlen v Laboratoriju za računalniške metode v elektroniki (LRME) na tej fakulteti, kjer se ukvarja z razvojem metod za umetno sintezo topologij analognih vezij ter z algoritmi za simulacijo elektronskih sistemov. Pedagoško sodeluje pri predmetih s področij programiranja, algoritmov in simulacij vezij. Njegova raziskovalna zanimanja obsegajo tudi področje informacijskih in izobraževalnih računalniških sistemov, spletnih tehnologij ter varnega zagona nezaupanja vredne programske kode v nizko privilegiranih okoljih.