HACK YOUR LIFE

BY PA1TONT4MI

Introduction to Hacking
By PaitonTami

Authors:
6ic4da
D1sM3
R00tM3
N3tR4t

თავი 0
დავამსხვრიოთ მითები ჰაკერებზე

სანამ ამ თემას შევეხებოდით აზრად მოგვივიდა(კონკრეტულად DisMels მოუვიდა ეს აზრი) ასეთი რამ: მოდი ხალხს გავაგებინოთ თუ რეალურად რა/ვინ არის ჰაკერი.რეჟისორებს ძალიან ცუდად აქვთ წარმოჩენილი ჰაკერები(გარდა სერიალისა “Mr.Robot”, რადგან სერიალში ნაჩვენები ჰაკინგ სცენები რეალობასთან საკმაოდ ახლოსაა, მაგრამ გარკვეულწილად მაინც გაზვიადებულია).ვთვლით, რომ სანამ ჰაკინგზე ვისაუბრებთ საჭიროა მკითხველმა გაიაზროს თუ რაზე ვსაუბრობთ.

* მითი 1: ჰაკერი არის კრიმინალი
  ყველა ჰაკერი კრიმინალი არაა.არსებობენ ჰაკერები, რომლებიც მთავრობისთვის მუშაობენ(ჩვენ მათ კიბერძაღლებს ვეძახით).

* მითი 2: თუ თქვენ თამაშში “cheat” გაქვთ თქვენ ჰაკერი ხართ
  გვაპატიეთ 8 წლის გივიკოებო და 13 წლის ნინჩოებო, მაგრამ თუ თქვენ მაგალითად “PUBG Mobile”ში გაქვთ ეგრედწოდებული “AimBot” ან “PhantomWall” ეს იმას არ ნიშნავს, რომ თქვენ ჰაკერი ხართ.თქვენ უბრალოდ თამაშის სერვერს “აბოლებთ”.ჰაკინგი საერთოდ სხვა რამაა.

* მითი 3: ჰაკერები სხედან ბნელ ოთახში ლეპტოპთან და წერენ მწვანე კოდებს
  ხშირად ჰაკერები კომპიუტერებს საერთოდ არ ეკარებიან.ჰაკერი მანიპულირებს ადამიანებით(ამაზე ვისაუბრებთ წიგნში).აქედან გამომდინარე კარგი ფსიქოლოგი ნიშნავს კარგ ჰაკერს(რაღაც დოზით).

* მითი 4: ჰაკერები ნიღბებით დადიან გარეთ
  ჩვენც ადამიანები ვართ.გთხოვთ ასეთ სისულელეებს ნუ დაიჯერებთ.

* მითი 5: ჰაკერები გამოდიან სიტყვით და ვიდეოში იცვლიან ხმას
  ავიღოთ ჰაკერული დაჯგუფება “Anonymous”.ჩვენ მათ კიბერცირკს ვუწოდებთ, რადგან არც ერთი ჰაკერი, რომელსაც 0.001 გრამი ტვინი მაინც უგდია თავში არ გადაიღებს სულელურ ვიდეოს და არ განათავსებს სოციალურ ქსელში.

* მითი 6: 12-13 წლის ბიჭები და გოგონები პენტაგონს და ნასას ტეხავენ
  ძვირფასო საზოგადოებავ.მოგმართავთ ჰაკერული კლანი “PaitonTami”(ანუ ჩვენ), რადგან ვერ ერკვევით აიტი საკითხებში ეს იმას არ ნიშნავს, რომ ყველა ადამიანის ნათქვამი უნდა დაიჯეროთ ვინც კი აიტიზე რაიმეს დააბრეხვებს.გვაპატიეთ, მაგრამ არ არსებობენ კომაროველი/ვეკუელი/ჰარვარდელი/ოქსფორდელი გენიოსები,

რომლებიც 15 წუთში სისტემას ტეხავენ, რომლის აწყობასაც დაახლოებით 8 თვე მოუნდა ფედერალური ბიურო.აქედან გამომდინარე თქვენი შვილი არაა ვუნდერკინდი მაშინაც კი, თუ მან საჯარო ბიბლიოთეკის შიდა ქსელი გატეხა.

მგონი მითები საკმარისად დავამსხვრიეთ(DisMe გვეხვეწებოდა კიდევ მაქვს 291 მითიო მაგრამ ხომ ხვდებით...ქალებს რას გავუგებთ).ახლა განვიხილოთ თუ როგორია რეალურად ჰაკერი.ჰაკერი არის ადამიანი, რომელიც ერკვევა კომპიუტერებში და ქსელებში.ის ამ ცოდნას ბოროტულად იყენებს.პარალელი გავავლოთ სპეცრაზმზე.სპეცრაზმელმა იცის იარაღის ხმარება, ჩასაფრება, შენობის აღება და ა.შ. მაგრამ ის ცუდად არ იყენებს ამ ცოდნას.მაგრამ არის ტერორისტი, რომელიც ამ ცოდნას ცუდად გამოიყენებს.ანუ ჩვენ ამით იმის თქმა გვინდა, რომ “ჰაკინგიც” არის ჩვეულებრივი skill და არა რაღაც სასწაული ეზოთერული ცოდნა, რომელიც ღმერთმა მხოლოდ ნამდვილ ქრისტიანებს გვარგუნა ანტიქრისტეს დასამარცხებლად.

ახლა ვუპასუხოთ ხშირად დასმულ კითხვებს(ამის თქმის შემდეგ დასმულებს).

* კითხვა 1: როგორია ჰაკერული ოპერაცია?

გააჩნია სამიზნეს.თუ სამიზნე მაგალითად არის რაღაც ოფისი სადაც აქვთ სერვერი, ჩვენ შეგვიძლია ქსელში ჩავუმონტაჟოთ ე.წ. “Raspberry Pi”, რომელიც ქსელის ტრაფიკს ჩაიწერს და მაგალითად ყოველ 1 საათში .pcap ფაილს ჰაკერის FTP(File Transfer Protocol) სერვერზე გააგზავნის.ეს საკმაოდ რთული ოპერაციაა.ჰაკერმა უნდა მოახერხოს და შენობაში “Raspberry Pi” ისე უნდა შეაპაროს და შემდეგ ქსელზე მიაერთოს, რომ ვერ გამოიჭირონ.შემდეგ ეს მოწყობილობა MITM შეტევით ჩაიწერს ქსელზე მიერთებული ყველა მოწყობილობის ტრაფიკს(მათ შორის იმ სერვერისას, რომელიც ჩვენი სამიზნე იყო).შემდეგ ჰაკერი ილოცებს, რომ ვიღაც სულელმა სერვერზე “telnet”ით login გადაწყვიტოს, რადგან “telnet”ის ტრაფიკი დაუშიფრავია და MITM შეტევით ამოვიკითხავთ სერვერის loginსა და passwordს(ახლა რასაც ვბოდიალობთ არაა სავალდებულო რომ გაიგოთ...ამ მომენტში...).

* კითხვა 2: რატომ იყო ამერიკის ტელევიზიაში რეპორტაში 13 წლის ჰაკერზე?

ალბათ ეს კითხვა იქიდან წამოვიდა რომ ვთქვით პატარა ბავშვი ჰაკერები არ არსებობენმეთქი.ჩვენი აზრით ასეთი რეპორტაჟები დადგმულია ან ბავშვს ტექსი აქვს დაზეპირებული და იმას ლაპარაკობს.არ გვესმის 11 წლის ბიჭმა საიდან უნდა იცოდეს ის, რომ IPv4 მისამართი არის 32 ბიტიანი.აიტი ზერთული სფეროა.სპეციალისტები წლები უნდებიან ამ სფეროს შესწავლას და ალოგიკურია, როდესაც ვიღაც ჟურნალისტი გამოხტება და ამბობს, რომ 11 წლის ბიჭმა პენტაგონის სერვერი დააექსპლოიტა.

ახლა წესით საბოლოო კითხვა უნდა ისმოდეს: ვეღარ გავიგე ჰაკერი რას აკეთებს, რა უაზროთ ხსნით?

პასუხი შემდეგია: ჰაკერი ქსელში ან ლოკალურ კომპიუტერზე ეძებს მოწყვლადობას ან არასწორად დაკონფიგურირებულ სისტემას და ის ამ შეცდომას სათავისოდ იყენებს.აი ეს არის ჰაკერის განმარტება.

ახლა მგონი დროა ვისაუბროთ იმაზე თუ ამ წიგნით როგორ ვაპირებთ აიტის საფუძვლების სწავლებას.ჩვენ ბევრი ვიფიქრეთ და გადავწყვიტეთ, რომ ავდგეთ და მთელი საბაზისო ცოდნა ამ წიგნში ჩავტენოთ.არ ვიცით ეს აზრად რატომ მოგვივიდა, ალბათ იმიტომ, რომ ვატყობთ თუ რამდენად არასწორად იგებს საზოგადოება ჰაკერის დეფინიციას.თავიდან ვთვლიდით, რომ ეს ჩვენი პრობლემა არ იყო, მაგრამ ახლა როდესაც ადამიანი გაიგებს, რომ ვიღაც ჰაკერია მას ჰგონია, რომ ის ან თამაშებში cheatებს იყენებს, ან ყოველ ღამე კაპიშონს წამოიხურავს, ნიღაბს გაიკეთებს და ეულად ქუჩაში დადის, რათა ბანკში ან კაზინოში შეიპაროს, გატეხოს უსაფრთხოების სისტემები და კამერები, შემდეგ მილიონი დოლარი მოიპაროს, გადარიცხოს ოფშორულ ანგარიშზე, გაიქცეს ქვეყნიდან, იყიდოს მაიამიში კერძო სახლი და დარჩენილი ცხოვრება ბედნიერად გაატაროს.ეს მითები ზევით უკვე დავამსხვრიეთ და ახლა დროა უკვე სწავლებაზე გადავიდეთ არა?

თავი 1
Introduction to Computers

სანამ პენტაგონს დავეტაკებით, მოდით მანამდე გავიაზროთ თუ როგორ მუშაობს კომპიუტერი.ალბათ ხშირად გიფიქრიათ ნეტავ რა დევს ამ ყუთში? როგორ მეძლევა საშუალება, რომ სოციალურ ქსელებში ყოფილებზე ვიჭორაო ან ონლაინ კაზინოშს თირკმელი მივყიდო იმის იმედით, რომ მილიონი freespinდან ერთი გამივარდება და მოვიგებ?პასუხი არც შავი მაგიაა და არც ზეკომპლექსური კვანტური მექანიკა.პასუხი ბევრად ადვილია.ადამიანებმა ელექტროობაზე ზემოქმედება ისწავლეს.სანამ ამას ავხსნით მანამდე ჩამოვთვალოთ თუ რა ნაწილებია საჭირო კომპიუტერისთვის.

1) MotherBoard – დედაპლატა

დედაპლატა არის PCB დაფა, რომელიც ყველა დანარჩენ ნაწილს “აერთიანებს”.ის იღებს კვების წყაროდან ელექტროენერგიას და ანაწილებს კომპიუტერის სხვადასხვა ნაწილზე.ასევე მასზეა მოთავსებული BIOS(Basic Input Output System) ან თუ შედარებით ახალი თაობის კომპიუტერი გაქვთ UEFI(Unified Extensible Firmware Interface).დედაპლატაზე ერთდება SSD/HDDს SATA კაბელები, კვების ბლოკის(ესეც კომპიუტერის ნაწილია და მალევე ვახსენებთ) კაბელები და ა.შ.

2) CPU(Central Processing Unit) – პროცესორი

პროცესორი პატარა კვადრატული ფორმის “დაფაა”, რომელიც დედაპლატაზე მდებარე სოკეტში(socket) ჯდება.პროცესორი არის

მთავარი გამომთვლელი კომპიუტერისთვის(შეგვიძლია მას კომპიუტერის “ტვინიც” ვუწოდოთ).

3) RAM(Random Access Memory) – ოპერატიული მეხსიერება
ოპერატიული მეხსიერება არის არაკონსტენტური მეხსიერება(ანუ იქ სამუდამოდ ვერ შევინახავთ მონაცემებს).მას იყენებს პროცესორი, რათა შეინახოს მისთვის საჭირო მონაცემები(ამასაც განვიხილავთ).

4) Graphics Card – ვიდეოკარტა
რატომღაც 6ic4da და N3tR4t თვლიან, რომ ვიდეოკარტა რაღაცმხრივ ცალკეული კომპიუტერია(ხუმრობით), რადგან მას აქვს საკუთარი პროცესორი ანუ GPU(Graphical Processing Unit) და VRAM(Video Random Access Memory).ვიდეობარათის მუშაობის პრინციპი საკმაოდ ადვილია.ვიდეოკარტაზე მყოფ გრაფიკულ პროცესორს(GPU) აქვს ერთი დავალება(ნუ ერთი არა, მაგრამ მთავარი ესაა): ეკრანზე გამოიტანოს გამოსახულება(დაარენდეროს 3D ობიექტები თამაშის დროს ან უბრალოდ ფილმს უყუროთ, რადგან მარტო ფილმის ხმა მოსაწყენი იქნებოდა).ამ პროცესორსაც სჭირდება RAM(ზევით ვახსენეთ), ამიტომაც მასზე დატანილია GPUსთვის განკუთვნილი RAM ანუ VRAM.

5) HDD(Hard Disk Drive) – მყარი დისკი ან ვინჩესტერი
ვინჩესტერი(შოთგანი ბუხ ბუხ) არის საწყობი(storage).აქ ინახება ფაილები, თამაშები, ყველა პროგრამა რასაც გადმოიწერთ და დააყენებთ.

5.5) SSD(Solid State Drive) – უბრალოდ სსდ დაუძახეთ
იგივე ფუნქცია აქვს, რაც HDDს უბრალოდ განსხვავებულადაა აგებული და ბევრად სწრაფია ვიდრე HDD(ანუ მაგალითად ვინდოუსი SSDზე უფრო მალე ჩაირთვება ვიდრე HDDზე).

6) PSU(Power Supply Unit) – კვების ბლოკი
ეს ნაწილი შტეფცელიდან იღებს ელექტროენერგიას და ანაწილებს კომპიუტერზე.

მგონი გასაგებია თუ რა ნაწილები დევს თქვენს ყუთში.მაგრამ აქ ერთ უცნაურობას ვაწყდებით.ზოგ კომპიუტერში არ დევს ვიდეოკარტა, მაგრამ ეკრანზე გამოსახულება მაინც გამოდის.ეს როგორ ხდება?ძალიან ადვილად.პროცესორის მწარმოებელმა კომპანიებმა(Intel და AMD) თავიანთ CPUებში ჩააშენეს GPUები.ასეთ გრაფიკულ პროცესორებს ეძახიან(ინტეგრირებულ გრაფიკულ პროცესორებს).მათ არ გააჩნიათ VRAM, შესაბამისად ის და CPU RAMს იყოფენ.

კარგი, კარგი.კი გაიგე თუ რა ნაწილები გიდევს შავ ყუთში(ან რა ფერიცაა შენი ქეისი), მაგრამ ახლა გაინტერესებს ეს ნაწილები როგორ მუშაობენ.მოდით ეგეც განვიხილოთ.სანამ ამის ახსნას დავიწყებთ ნანამდე ცოტა ელექტროობასაც უნდა შევეხოთ.რა არის დენი?დენი არის პატარა უარყოფითად დამუხტული ბურთულების ნაკადი.მათ ელექტრონები ეწოდებათ.თუ ელექტრონი არის, მაშინ ვწერთ 1ს, ხოლო თუ ელექტრონი არ

არის მაშინ ვწერთ 0ს(უფრო ტექნიკურად რომ ვთქვათ 1 ნიშნავს შედარებით მაღალ ძაბვას, ხოლო 0 ნიშნავს შედარებით დაბალ ძაბვას).ამ ელექტრონებზე მანიპულაციით ვიღებთ საოცარ შედეგებს(აი შენ რომ csgoში გაქვს 200fps მაგ შედეგზე ვლაპარაკობთ).პროცესორს ზედ აქვს უამრავი flip-flop(გადამრთველ-გადმომრთველი აი ისეთი შუქს რომ რთავ და თიშავ ოთახში) და უამრავი ლოგიკური გეითი(Logical Gate), როგორიცაა მაგალითად: AND, OR, NOT, NOR, XOR, NAND და ა.შ.(ლოგიკურ გეითებს ნუ აგვახსნევინებთ ინტერნეტში მოძებნეთ გთხოვთ).ამ გეითებში ელექტრონების გატარების შედეგად ვიღებთ რაღაც შედეგებს.მაგალითად სულ რამდენიმე ლოგიკური გეითით შეიძლება 8 ბიტიანი კალკულატორის აწყობა.

კარგი ისიც გასაგებია მგონი თუ დენი სად და როგორ გადის, მაგრამ ახლა ალბათ დაგაინტერესათ თუ რაზე ხდება ეს ყველაფერი ანუ რა მატერიაა მაგალითად დედაპლატა და ვიდეოკარტა.ასეთ მწვანე/შავ/ლურჯ პლატებს/დაფებს ეძახიან PCBებს(Printed Circuit Board).ასეთი დაფები მუშაობს შემდეგნაირად: მათზე დატანილია ძალიან პატარა და წვრილი სპილენძის(Copper, პერიოდულ სისტემაში სიმბოლო “Cu”, ატომური ნომერია 29, არის კარგი გამტარი) “ხაზები”, რომლებსაც გადააქვთ ელექტრონები ერთი ადგილიდან მეორეზე(მაგალითად კვების ბლოკიდან კაბელი ერთდება დედაპლატაზე.ამ კაბელიდან ელექტრონი გადადის დედაპლატაზე და შემდეგ დედაპლატა ამ ელექრონებს აწვდის პროცესორს).

ბევრი უაზრო ბოდიალის შემდეგ ვფიქრობთ, რომ კომპიუტერის ფიზიკური მხარის(Hardware) რაღაც ცოდნა ჩაგიტენეთ თავში.ახლა კი განვიხილოთ “სულიერი”(ვხუმრობთ) მხარე(Software).

* უცებ წამოსული კითხვა: რა არის Hardware და Software?
    Hardware არის კომპიუტერის ნაწილები(პირდაპირი გაგებით).
    Software არის პროგრამები, რომლებსაც Hardwareზე ვუშვებთ(GTA, Fortnite, Minecraft, Windows, Linux, BIOS, Google Chrome, Firefox, Graphics Card drivers).

როგორ მუშაობს Software?ამის ახსნა საკმაოდ ადვილია.არიან ადამიანები, რომლებსაც პროგრამისტებს უწოდებენ(ეს ის პროგრამისტი არაა ვინდოუსს რომ 10 ლარში გიყენებს და კომპიუტერს “ქოქავს”).პროგრამისტები კომპიუტერებს აპროგრამებენ(ანუ პროცესორს ინსტრუქციებს აძლევენ).კომპიუტერის დაპროგრამება ხდება კოდის წერით.დასრულებულ კოდს რომ გავუშვებთ ირთვება რაღაც პროგრამა(Software).

თავი 2
Introduction to Internet

მოდით ვისაუბროთ ინტერნეტზე.რა არის ინტერნეტი?ინტერნეტი არის კავშირის საშუალება(არა, ის არ მოუგონიათ მასონებს რათა

გაკონტროლონ.უბრალოდ გვაცადეთ ახსნა).ინტერნეტი მოიგონეს იმისათვის, რომ ერთი მოწყობილობიდან მეორეზე მონაცემები გაეგზავნათ(ვიდეო სტრიმინგი, მესიჯები, ტელეკომუნიკაცია, მეილები, ონლაინ თამაშები, საიტები, მეტავერსი და ა.შ.).ინტერნეტში ყველა მოწყობილობას აქვს მისამართი.მისამართების გარეშე მოწყობილობები ვერ გაიგებენ ვის უნდა ეკონტაქტონ.ამ მისამართებს ეწოდებათ IP(Internet Protocol) მისამართები.IP მისამართი არის ორნაირი(და კიდევ ორნაირი ოღონდ სხვანაირად, მერე ვიტყვით): IPv4 და IPv6.სანამ იმას ვიტყვით თუ რა სხვაობაა ამ ორს შორის, მანამდე მოდით გავიაზროთ თუ როგორ მუშაობს IP მისამართები.ავიღოთ მაგალითად IPv4 მისამართი:

192.168.1.1

პირველად იმას ვაკვირდებით, რომ IPv4 მისამართი ჩაიწერება 4 რიცხვით.ახლა უნდა გავიაზროთ რა ლიმიტები აქვს მას და რატომ.IPv4 მისამართში რიცხვები შეიძლება იყოს 0დან 255ის ჩათვლით.რატომ?ეს არის ორობითი თვლის სისტემის ბრალი.ზემოთ ხომ ვახსენეთ, რომ პროცესორს 1 და 0 ესმის მხოლოდ.ანუ ჩვენ მხოლოდ ამ ორი ციფრით თვლა შეგვიძლია.ადამიანებს გვაქვს ათობითი თვლის სისტემა:

0 1 2 3 4 5 6 7 8 9

მაგრამ კომპიუტერს აქვს ორობითი თვლის სისტემა:

0 1

ახლა გავიაზროთ თუ როგორ ითვლიან კომპიუტერები.ადამიანები ვითვლით ასე: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9(აი აქ უკვე აღარ გვყოფნის სიმბოლოები, ასე რომ ჩვენ გვერდით ვუწერთ სიმბოლოს), 10.
ახლა დავაკვირდეთ კომპიუტერისას: 0, 1(აი აქ უკვე აღარ გვყოფნის სიმბოლოები, ამიტომ რაღაც უნდა მოვიფიქროთ).არც ისე ბევრი ფიქრის შემდეგ ადამიანები მიხვდნენ, რომ ორობითში თვლაც შესაძლოა:

0, 1, 10, 11, 100, 101, 110, 111, 1000

ცოტა რთული ჩანს არა?ჩანს მაგრამ ადვილია.დააკვირდით 1ის შემდეგ 10 წერია.ეს იმიტომ, რომ 1 ბოლო ციფრია ორობით სისტემაში(როგორც 9 ათობითში) და ამიტომ ჩვენ გვერდით ვუწერთ ნულიანს.10ის შემდეგ წერია 11, ეს იმიტომ, რომ 0ის ადგილას შეგვიძლია 1 დავწეროთ.11ის შემდეგ მოდის 100, ეს იმიტომ, რომ 11ში ბოლო 1ის ადგილას ვეღარფერს დავწერთ, ამიტომაც ორივე 1 ნულდება და მარცხნივ ეწერება ახალი 1.თუ ჩვენი ახსნილი ვერ გაიგეთ გირჩევთ youtubeში მოძებნოთ: Binary explained.აუცილებლად იპოვით 2 წუთიან ვიდეოს სადაც საოცრად მარტივად აგიხსნიან ყველაფერს.

სანამ IP მისამართებს დავუბრუნდებით მოდით განვიხილოთ მონაცემთა სიდიდეები.1 ბიტი(Bit) ნიშნავს 1 ელექტრონს.8 ბიტი(Bit) კი ნიშნავს 1 ბაიტს(Byte).ქვემოთ მოცემულ ტოლობებს დააკვირდით და ყველაფერს მიხვდებით:
1 Bit = 1 ელექტრონი
8 Bit = 8 Byte
1 000 Byte = 1KB

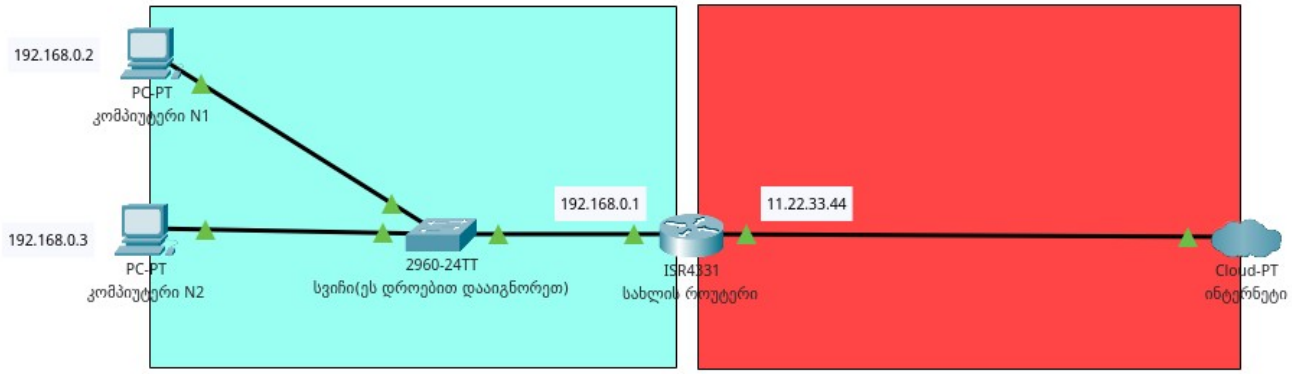
1 000 KB = 1 MB
1 000 MB = 1 GB

ახლა დავუბრუნდეთ IPv4 მისამართს.რატომ არის ამ მისამართში რიცხვის ზომა მაქსიმუმ 255?იმიტომ, რომ IPv4 მისამართში თითო რიცხვს აღწერს 1 Byte(8 Bit).ანუ ჩვენ 8 ელექტრონით გვიწევს ჩაწერა ამ რიცხვების.მაგალითად:

255.255.255.0
11111111.11111111.11111111.0

იმედია გასაგებად ავხსენით.8 ერთიანი ნიშნავს 255ს და რადგან 8 ბიტის მეტი არ გვაქვს 256საც ვერ დავწერთ რადგან მეცხრე ბიტისთვის ადგილი აღარაა.ამიტომაა, რომ IPv4ში გვაქვს რიცხვთა მაქსიმუმი სიდიდე 255.

გავაგრძელოთ ინტერნეტზე საუბარი.ამ მისამართებით ხდება კომუნიკაცია ინტერნეტში.მაგალითად google DNS(Domain Name System) სერვერის IPv4 მისამართია 8.8.8.8, ხოლო თქვენი მისამართია 11.22.33.44 და თქვენ გაინტერესებთ ვებსაიტ “somewebsite.com”ის მისამართი.თქვენ 11.22.33.44 მისამართიდან აგზავნით მოთხოვნას 8.8.8.8ზე და ეკითხებით DNS სერვერს თუ რა არის ზემოთხსენებული საიტის IPv4 მისამართი.googleის სერვერს დამახსოვრებული აქვს თქვენი მისამართი, რათა პასუხი თქვენ მოგაწოდოთ და არა ვიღაც სხვას(მისამართების გარეშე ეს ხომ შეუძლებელი იქნებოდა) და პასუხს თქვენ გიბრუნებთ.თუ ვერ გაიგეთ მისამართების მუშაობის პრინციპი მოდით ასეთ მაგალითს მოვიყვანთ:
მაგალითად თქვენ ცხოვრობთ კომუნისტურად მოაზროვნე სამეზობლოში და ყველამ გადაწყვიტეთ, რომ ყველა მეზობლის სახლი, მანქანა, კარები, ეზო, ღობეები იქნება ერთნაირი.მხოლოდ მისამართები გაქვთ სხვადასხვა.მაგალითად: იოსებ სტალინის N31.თქვენ წვეულებას აწყობთ სახლში და გინდათ, რომ სხვა ქალაქიდან თქვენი მეგობარი გეწვიოთ.თქვენს მეგობარს ეუბნებით თქვენსავე მისამართს და ის მოდის ამ მისამართზე.მგონი ლოგიკურია არა?ასეა ინტერნეტშიც.მისამართებს რიცხვებით აღვწერთ, რომ სხვადასხვა მოწყობილობებთან კომუნიკაცია შედგეს.მისამართების გარეშე ვერავინ გაიგებდა ვინ ვის ეკონტაქტება და სრული ქაოსი იქნებოდა(კი არადა საერთოდ ვერ შედგებოდა კონტაქტი ვერავისთან).კარგით ახლა განვიხილოთ კერძო(Private) და საჯარო(Public) IP მისამართები(არა, კერძო მდიდარის და საჯარო ღარიბის არ არის).რას ნიშნავს კერძო და საჯარო მისამართები?



აქ გვაქვს ძალიან კარგი მაგალითი იმისა, თუ რას ნიშნავს კერძო და საჯარო IP მისამართები.წითელი ზონა ნიშნავს ინტერნეტს(WAN – Wide Aria Network), ხოლო ცისფერი ზონა ლოკალურ ქსელს(LAN – Local Aria Network).უფრო ადვილად რომ ავხსნათ ასეთ მაგალითს მოვიყვანთ: ინტერნეტი არის ქუჩა, ხოლო ლოკალური ქსელი არის სახლი.ინტერნეტი(WAN) საჯაროა, ხოლო ლოკალური ქსელი(LAN) კერძო.როგორც ხედავთ სახლის როუტერს ორი IPv4 მისამართი აქვს.პირველია 192.168.0.1(კერძო), ხოლო მეორეა 11.22.33.44(საჯარო).საიტებზე, თამაშების სერვერებზე და ა.შ. გამოიყენება საჯარო IP მისამართი.შიდა ქსელში კომუნიკაციისათვის გამოიყენება კერძო IP მისამართები.

* როგორ გავარჩიოთ საჯარო კერძოსგან?
    კერძო IP მისამართები ძალიან ცოტაა.ის ორნაირია:
    1) 192.168.0.0
    2) 10.0.0.0
    ანუ თუ მისამართი იწყება “10”ზე ან “192.168”ზე ესეიგი ის კერძოა, დანარჩენი ყველა კი საჯარო.

არის მესამე სახის IP მისამართიც.ეს გახლავთ:
127.0.0.1

რა არის ეს?ეს არის Localhost(ლოკალური ჰოსტი).მაგალითად თქვენ ხართ web დეველოპერი და თქვენი კომპიუტერით მუშაობთ ვებსაიტზე.საშინელება იქნებოდა, რომ იძულებული ყოფილიყავით აგეღოთ მეორე მოწყობილობა და გამოგეყენებინათ ტესტ სერვერად.ამისათვის მოიგონეს Localhost.ამ მისამართზე თუ რაიმეს გაუშვებთ ის თქვენივე კომპიუტერზე გახორციელდება.Localhostთან კონტაქტი წარმოიდგინეთ როგორც საკუთარ თავთან ლაპარაკი(ოღონდ ჩუმად, ტვინში).

თავი 3

ინტერნეტი უფრო დეტალურად

ინტერნეტში და ზოგადად ქსელში საკონტაქტოდ რაღაც პროტოკოლებს ვიყენებთ.ახლა უნდა განვიხილოთ ერთ-ერთი მოდელი(OSI).ამ მოდელით კარგად აღიქმება თუ როგორ ხდება ორ მოწყობილობას შორის კავშირი.

დავაკვირდეთ მოდელს და განვიხილოთ ყველა ფენა(layer).

1) Physical(ფიზიკური ფენა) – ეს ფენა ნიშნავს პირდაპირი გაგებით Hardwareს.ანუ კაბელებს, ჰაბს, კომპიუტერში ინტერნეტის კაბელის შესაერთებელს(პორტს) და ა.შ.

2)Data Link(მონაცემთა გადაცემის არხის ფენა) – ამ ფენის დავალებაა, რომ უზრუნველყოს ქსელურ მოწყობილობებს შორის მონაცემთა გადაცემა და პირველ ფენაზე მომხდარი შეცდომების აღმოფხვრა.ამ ფენაზე კავშირი ხდება MAC(Media Access Control) მისამართებით, რომლებიც აპარატულ ნაწილში ფიქსირდება(წარმოების დროს წერენ მასში ამ მისამართს).

3) Network(ქსელური ფენა) – ეს ფენა უზრუნველყოფს მონაცემების ყაროდან(source) დანიშნულების(destination) ადგილამდე მიტანას ერთი ან რამდენიმე ქსელის გავლით.

4) Transport(ტრანსპორტული ფენა) – ეს ფენა უზრუნველყოფს მონაცემების ეფექტურ გადატანას.

5) Session(სესსის ფენა) – ეს ფენა აკონტროლებს დიალოგს ქსელურ მოწყობილობებს შორის.ის ამყარებს ან წყვეტს კავშირს სხვადასხვა მოწყობილობებთან.

6) Presentation(პრეზენტაციის ფენა) – ეს ფენა გარდაქმნის მონაცემებს აპლიკაციური ფენისთვის გასაგებ ენაზე.

7) Application(აპლიკაციის ფენა) – ამ ფენით მომხმარებელს შეუძლია ქსელში მყოფ მონაცემებზე წვდომის აღება.

თავიდან ეს ცოტა ჩახლართული შეიძლება მოგეჩვენოთ, მაგრამ დროდადრო გაუგებთ.დღესდღეობით OSI მოდელი აღარ გამოიყენება და მას უბრალოდ მაგალითისთვის იყენებენ.დღესდღეობით გამოიყენება TCP(Transmission Control Protocol) კომუნიკაციისათვის.ეს პროტოკოლი უზრუნველყოფს მონაცემთა ზუსტ გადატანას(დარწმუნდება, რომ მონაცემები დანიშნულების ადგილამდე მივიდა.

ახლა ვისაუბროთ თუ რა არის პორტები.პორტები რამდენიმე სახისაა.შესაძლოა ეს იყოს უბრალოდ USB პორტი, ან Ethernet(ინტერნეტის შესაერთებელი) პორტი, ან ყურსასმენის პორტი, მაგრამ ჩვენ აქედან არც ერთი არ გვაინტერესებს.ჩვენ საუბარი გვაქვს არაფიზიკურ პორტებზე.ეს პორტები მდებარეობს მეშვიდე ფენაზე(ზევით წაიკითხეთ თუ არ იცით რაზეა საუბარი) ანუ რაღაც აპლიკაციები აღებენ თქვენს მოწყობილობაზე ამ პორტებს.წარმოიდგინეთ პორტი როგორც კარები.მაგალითად youtube.comს აქვს HTTPS(Hyper Text Transfer Protocol Secure) გახსნილი, რათა მიიღოს კლიენტები(ანუ თქვენ) და აყურებინოს ვიდეოებს.ანუ ეს არის ვებ სერვერი(ვებსაიტია დაჰოსტილი).თქვენ ამ “კარებში” შედიხართ და აღმოჩნდებით youtube.com ვებსაიტზე.ანუ პორტი არის რაღაც სერვისი მომხმარებლისთვის.მაგალითად გვაქვს SSH(Secure Shell) პორტი გახსნილი კომპიუტერზე.მეორე კომპიუტერს შეუძლია ამ პორტს დაუკავშირდეს და ეს პორტი მას წვდომას მისცემს იმ კომპიუტერის CLIზე(Command Line Interface), რომელმა კომპიუტერმაც გახსნა SSH პორტი.მაგრამ სავალდებულო არაა პორტზე რაიმე სერვისი იყოს.ასეთ პორტებს ჩვენ უმ(raw) პორტებს ვეძახით.მათი გახსნა ასეა შესაძლებელი:

nc -l 11

მაგალითისთვის ავიღეთ მე-11 პორტი.ამ პორტზე შემოსული კლიენტი ვერაფერს მიიღებს, რადგან პორტი არაფერს ემსახურება.ის უბრალოდაა გახსნილი და ელოდება შემომავალ კავშირებს.

რადგანაც პორტები განვიხილეთ საუბარი TCPზე განვაგრძოთ.არსებობს TCP პორტები.ეს იმას ნიშნავს, რომ ამ პორტებზე მოქმედებს TCP პროტოკოლი.თვითონ TCP, როგორც უკვე ზევით ავღნიშნეთ უზრუნველყოფს მონაცემთა ზუსტ გადატანას ერთი მოწყობილობიდან მეორეზე.

* ამას როგორ ახერხებს?

ამას ახერხებს “three way handshake”ის დახმარებით.

* რა არის three way handshake?

ეს არის მეთოდი, რომლის დახმარებითაც TCP კავშირს ამყარებს ორ მოწყობილობას შორის.

ავხსნათ თუ როგორ ხდება ეს.ვთქვათ: თქვენი კომპიუტერიდან გინდათ შეხვიდეთ www.facebook.comზე.თქვენი კომპიუტერი გააგზავნის მოთხოვნას HTTPS პორტზე.თქვენი კომპიუტერი ამ მოთხოვნაში რამოდენიმე რაღაცას ჩაწერს.ამათგან ყველაზე მნიშვნელოვანია: წყარო IP, დანიშნულების IP, წყარო პორტი, დანიშნულების პორტი, seq number.
რა არის seq number?ეს არის TCPს მიერ დაგენერირებული რიცხვი მაგალითად 200 001.ამ რიცხვს გაუგზავნის დანიშნულების ადგილს(ამ შემთხვევაში www.facebook.com), როდესაც ეს მონაცემები(პაკეტი) მივა დანიშნულების ადგილამდე ის გახსნის ამ მონაცემებს(პაკეტს) და წაიკითხავს თუ რა წერია შიგნით.ანუ მან უკვე იცის თქვენი IP(წყაროს მისამართი), ის პორტი საიდანაც გაგზავნეთ ეს მონაცემები და თქვენს მიერ გაგზავნილი seq რიცხვი.ის უკან დაგიბრუნებთ გარკვეულ მონაცემებს(რაც საჭიროა facebook.comზე გადასასვლელად), თქვენს seq რიცხვს მიუმატებს 1ს და ისე დაგიბრუნებთ უკან(200 002) და ამასთანავე გამოაგზავნის საკუთარ seq რიცხვს მაგალითად 5001 და დაელოდება როდის უპასუხებთ.როდესაც მოვა ეს მონაცემები(პაკეტი) და თქვენი კომპიუტერი წაიკითხავს ამ მონაცემებს, ის აღმოაჩენს, რომ სერვერმა საკუთარი seq რიცხვიც გამოაყოლა.თქვენი კომპიუტერი ადგება და მის seq რიცხვს დაუმატებს 1ს და უკან გაუგზავნის სერვერს, ანუ გაუგზავნის 5002ს.აი ასე, TCP კავშირი შედგა.

ჩვენ ასევე გვაქვს მეორე პროტოკოლიც.მას ეწოდება UDP(User Datagram Protocol).საქმე იმაშია, რომ ის არ უზრუნველყოფს მონაცემების ზუსტ მიწოდებას, მაგრამ ის არის უფრო სწრაფი, რადგან არ ხდება seq რიცხვების გაცვლა.UDP კავშირს ვიყენებთ თამაშების თამაშის დროს, პირდაპირი ეთერის ყურების დროს(მაგალითად Twitch პლატფორმაზე).ჩვენ რომ აქ TCP გამოვიყენოთ ეს პროცესი იქნებოდა ძალიან ნელი და არაკომფორტული(არაპრაქტიკულიც).

თავი 4
Introduction to Programming

ახლა კი ავხსნათ თუ რა არის და როგორ მუშაობს პროგრამირება.მარტივად რომ ვთქვათ, პროგრამირება არის კომპიუტერთან(კონკრეტულად პროცესორთან) საუბარი.პროგრამირების დროს ჩვენ მას ვაწვდით ინსტრუქციებს და ის მათ ასრულებს.არსებობს უამრავი პროგრამირების ენა(ისევე როგორც ადამიანთა ენებშია).მაგალითად: Python, C, C++, C#, Ruby, Java, Javascript, Lua, Rust, SQL, Perl და ა.შ.

* როგორ და სად ხდება კოდის დაწერა?
    კოდის დაწერა ხდება text editorში(vscode, nano, neovim, vim, sublime, atom).ტექსტური ედიტორი ხსნის კონკრეტულ ფაილს და მასში წერს კოდს.როდესაც მოვრჩებით კოდის წერას, ჩვენ ამ ფაილს გავუშვებთ და ის მოხდება, რაც კოდში გვიწერია.

ყველა პროგრამირების ენას აქვს ფაილის გაფართოება.მაგალითად, ჩვეულებრივი ტექსტური ფაილი იქნებოდა ასეთი:

file.txt

აქ “file” არის ფაილის სახელი(ნებისმიერი რამ შეიძლება იყოს), ხოლო .txt არის ფაილის გაფართოება.გაფართოებით კომპიუტერს ვეუბნებით თუ რა სახის ფაილია ის.მაგალითად ვიდეო ფაილი იქნებოდა file.mp4 და აუდიო ფაილი იქნებოდა file.mp3.პროგრამირების ენებსაც საკუთარი გაფართოებები აქვთ.მაგალითად:
Python – file.py
C – file.c
Javascript – file.js
Lua – file.lua
C++ - file.cpp
Bash(Shell scripting) – file.sh

პროგრამირების ენებში რამდენიმე განსხვავებაა: სინტაქსი, გაშვებადობა, და დონე.

1) სინტაქსი – ეს არის წერის სტილი.ყველა პროგრამირების ენას სხვადასხვანაირი წერის სტილი გააჩნია.

2) გაშვებადობა – ეს ნიშნავს იმას, თუ როგორ უნდა გაეშვას კოდი.არის ინტერპრეტარული და კომპილატორული მეთოდები.ინტერპრეტარული გაშვებადობა არის შემდეგნაირი: გვაქვს ინტერპრეტერი(Interpreter) და ამ ინტერპრეტერს ვაკითხებთ იმ ფაილს, სადაც წერია კოდი.შემდეგ ინტერპრეტერი თითოეულ ხაზს გადასცემს პროცესორს.
კომპილატორული მეთოდი ოდნავ განსხვავებულია.გვაქვს კომპილატორი(compiler) და მას ვაძლევთ იმ ფაილს სადაც წერია კოდი.შემდეგ კომპილატორი ამ ფაილს გადააკომპილირებს და outputის სახით მოგვცემს ახალ ფაილს სადაც იქნება machine code(მანქანური კოდი).შემდეგ გადაკომპილირებულ ფაილს გავხსნით და პროგრამა იმუშავებს.

* რა არის მანქანური კოდი?
მანქანური კოდი არის ის basic ინსტრუქციები, რომლებიც პროცესორმა უნდა შეასრულოს.ანუ კომპილატორი კოდს თარგმნის კოდს პირდაპირ მანქანურ ენაზე, ხოლო ინტერპრეტერი პირდაპირ ფაილიდან თარგმნის და აწვდის პროცესორს კოდს.

საქმე ისაა, რომ ყველა ენა არაა პროგრამირების ენა.არსებობს მარკაპ(Markup) ენებიც, რომლებსაც დიზაინისათვის იყენებენ.მაგალითად html და css არის ენები, რომლებითაც იწყობა საიტების წინა მხარე(Front-End).ანუ როდესაც თქვენ რაიმე ვებსაიტზე შედიხართ, ის თქვენ გაწვდით html და css ფაილებს, შემდეგ თქვენი ბრაუზერი ამ კოდებს კითხულობს და საიტს გირენდერებთ.

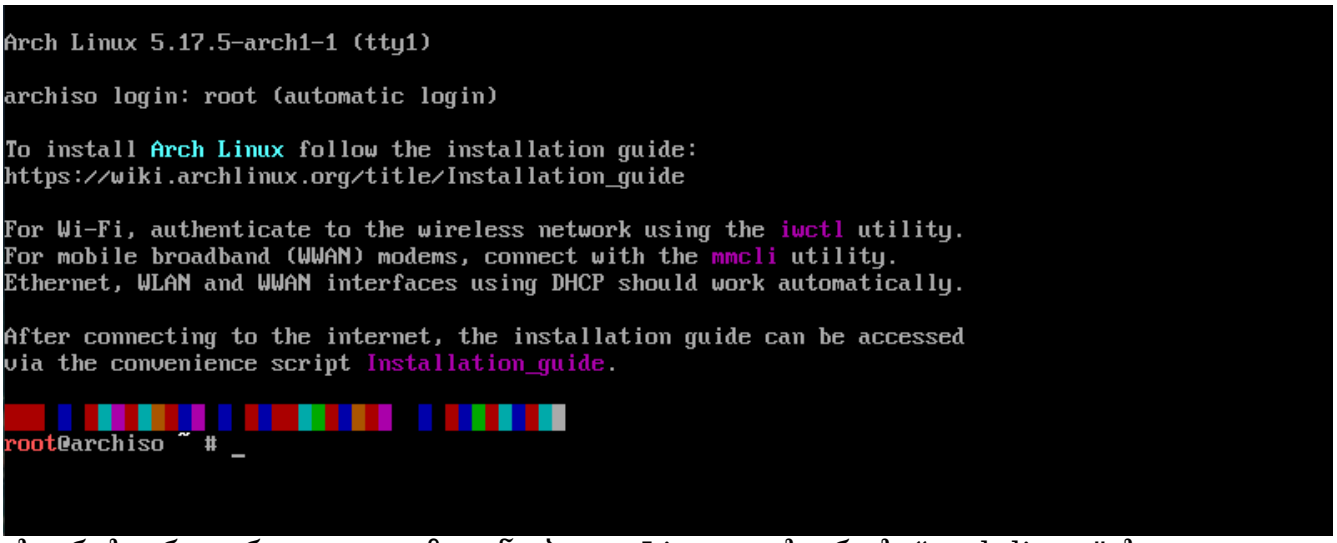
თავი 5
Introduction to OS

რა არის OS(Operating System)?ოპერატიული სისტემა არის პროგრამა, რომელიც ეშვება კომპიუტერის ჩართვისთანავე.ის უზრუნველყოფს მომხმარებლის კომფორტს.ოპერატიული სისტემებია: Windows, Linux(ნუ ეს რეალურად კერნელია(Kernel), მაგრამ მასზე ბევრი დისტრიბუტივია დაფუძნებული), MacOS, IOS, Android(ეს Linuxზეა დაფუძნებული), FreeBSD, Unix, Plan9 და ა.შ.

ოპერატიული სისტემები ერთმანეთთან აკავშირებს Hardwareს და Softwareს.ოპერატიული სისტემა Hardwareთან კომუნიკაციისათვის იყენებს დრაივერებს(driver).მაგალითად თქვენ გაქვთ ვიდეოკარტა “MSI Nvidia GeForce GTX 1080 ti”.ვინდოუსი მასთან საკონტაქტოდ გამოიყენებს დრაივერს.ყველა კომპონენტს სჭირდება დრაივერი.გაითვალისწინეთ, რომ კომპონენტი არ ნიშნავს კომპიუტერის ნაწილს.მაგალითად კომპიუტერის კომპონენტია ინტერნეტის პორტი(Ethernet port).იმისათვის, რომ ვინდოუსმა გამოიყენოს ეს ინტერნეტის პორტი საჭიროა კონკრეტულად ამ პორტის დრაივერი.

* რისგან შესდგება ოპერატიული სისტემა?

  ოპერატიული სისტემა შესდგება რამდენიმე მთავარი ნაწილისგან.ერთ-ერთი მთავარია Kernel.ეს არის ოპერატიული სისტემის ბირთვი.მაგალითად Linux ავიღოთ.მომხმარებელმა, რომ ლინუქსის კერნელთან კონტაქტი შეძლოს, ამისათვის სისტემა იყენებს syscallებს.ასევე გვაქვს Display managerები.რეალურად არც ერთ ოპერატიულ სისტემას არ აქვს GUI(Graphical User Interface).უბრალოდ არის სპეციალური პროგრამები, რომლებიც გვაძლევენ იმის საშუალებას, რომ გრაფიკული გამოსახულებები გვქონდეს კომპიუტერზე და არა მხოლოდ ტექსტური.



ეს არის ერთ-ერთი ყველაზე ცნობილი Linux დისტროს “arch linux”ის სქრინშოტი.რეალურად ოპერატიული სისტემა ასე გამოიყურება.მომხმარებელი ზის TTY(TeleTypewriter)ში.TTY იძლევა იმის საშუალებას, რომ ოპერატიულ სისტემას მივცეთ ბრძანებები(Commands).მაგრამ ჩვენ შეგვიძლია დავაყენოთ სპეციალური პროგრამები(Display Manager), რომლებიც GUIს გამოიტანენ ეკრანზე.ანუ ჩვენ შეგვეძლება სამუშაო გარემო გვქონდეს კომპიუტერში და არა მხოლოდ TTY.Display managerის ინსტალაციის შემდეგ საჭიროა უკვე Window Managerის ინსტალაცია.

* რა განსხვავებაა ამ ორს შორის?
    Display Manager უბრალოდ GUIს ქმნის ეკრანზე, ხოლო Window Managerს ევალება ფანჯრების გამოტანა, bar(მაგალითად PolyBar)ის დაყენება და ა.შ.

საბედნიეროდ არის საშუალებები, რომლებიც საშუალებას გვაძლევენ DMისა(Display Manager) და WMის(Window Manager) გარეშეც ვიმუშაოთ(N3tR3tისა და DisMeს თქმით ეს სრული საშინელებაა და მართლებიც არიან).

თავი 6
ცოდნის მიღებისგან შესვენება და ერთ-ერთი უცნაური თემის განხილვა(თუ გინდათ გადაახტით)

სამწუხაროდ კიდევ უამრავი სტერეოტიპი უნდა დავამსხვრიოთ.ჩვენ დიდი ხანია ვაკვირდებით საქართველოში მიმდინარე მოვლენებს, აიტი კუთხით და ძალიან სასაცილოა ის თუ საიდან მოიყარა ამდენმა სულელმა თავი ერთად.საქმე ისაა, რომ რატომღაც ყველა ადამიანს ჰგონია, რომ თუ ადამიანი ჰაკერია ის უეჭველად Linuxის მომხმარებელი უნდა იყოს, თან კონკრეტულად

“Kali Linux”ის მომხმარებელი.რა არის Kali Linux?ეს არის ლინუქსის ერთ-ერთი დისტრო, დაფუძნებული Debianზე(ესეც ლინუქსის დისტროა), რომელიც შექმნეს სპეციალურად Pen-Testingისათვის და “ეთიკური” ჰაკინგისთვის.ჩვენ ძალიან დიდი ხანია ვაკვირდებით და ვხედავთ, რომ საქართველოში უამრავი აკადემია/სასწავლებელი წამოვიდა, რომლებიც ამბობენ რომ ბავშვებს “ეთიკურ” ჰაკინგს ასწავლიან.ჩვენ გვინდა, რომ ეს თემა კარგად გავშალოთ.

მოკლედ, არის რამდენიმე სასწავლებელი facebookზე რომლებიც ყიდიან თავიანთ კურსებს “ეთიკურ” ჰაკინგზე.ჩვენ მათ სპეციალურად არ ვასახელებთ.პრობლემაა მათი მიდგომის ხერხები.მათ პროდუქტზე და გათვლაზე(ანუ რა მასაზე ცდილობენ ამის გაყიდვას) ეტყობათ არაპროფესიონალიზმი.პროდუქტი იმიტომ არ მოგვწონს, რომ ბავშვებს ასწავლიან ისეთ სისულელეებს როგორიცაა Githubდან სხვისი კოდების კლონირება და ამ კოდებით ძალიან სუსტი სამიზნეების დაექსპლოიტირება.ეგ ადამიანს არაფერში წაადგება.საერთოდაც ჰაკინგ სამყაროში ეგეთ ადამიანებს Script Kiddieებს ეძახიან, ანუ მარტივად რომ ვთქვათ ნუბებს.ნამდვილი ჰაკერი სხვის დაწერილ სისულელეებს არ გამოიყენებს სამიზნის დასაექსპლოიტირებლად.რა თქმა უნდა ბორბლის მეორედ გამოგონება საჭირო არაა და რაღაც მომენტში შესაძლოა სხვისი სკრიპტი გაუშვათ, მაგრამ მთელი ცხოვრება, რომ სხვის სკრიპტებს უშვებთ და არც კი იცით როგორ მუშაობს(და ხშირ შემთხვევაში სამიზნეს საერთოდ ვერ აზიანებთ) ეს ძალიან ცუდია.ამიტომაც დავიწყეთ ამ წიგნის წერა, რათა ადამიანებს სწორი გზა ვაჩვენოთ.რა თქმა უნდა არც ჩვენ ვართ პროფესიონალები, მაგრამ საკმაო გამოცდილება დაგვიგროვდა ამ სფეროში და ვიცით რასაც ვლაპარაკობთ(მათგან განსხვავებით).უცნაურია, როდესაც ადამიანმა Binary exploitationის საფუძვლებიც კი არ იცის, და ამბობს რომ ჰაკერია, მხოლოდ იმიტომ, რომ მისი კომპიუტერი Kali Linuxზე მუშაობს და არა Windowsზე ან Linux Mintზე.ჩვენ ვფიქრობთ, რომ ნამდვილი პროფესიონალური Linux დისტროები არის: Arch, Gentoo და Void.ეს იმიტომ, რომ ისინი მინიმალური დისტროებია და ყველაფერს შენით აკონფიგურირებ.მაგრამ ამ Kaliს ყველაფერი პრედაკონფიგურირებული მოყვება ანუ მომხმარებელმა არ იცის მის კომპიუტერში რა არის გაშვებული და რას აკეთებს.ეგეთი აკადემიებისგან კურსების ყიდვა ან დასწრება სრული სისულელეა, რადგან საბოლოოდ მაინც არაფერი დაგრჩებათ.ეს აკადემიები ასწავლიან ძალიან საწყის ეტაპებს.ისეთ საწყისებს, რომ ადამიანი, რომელიც ნორმალურად ერკვევა კომპიუტერებში იტყვის, რომ ასეთ კურსში ფულის ყრა ნამდვილი სისულელეა(ისიც დაიმახსოვრეთ, რომ რაშიც ისინი ფულს ითხოვენ youtubeზე უფასოდ დევს).

თავი 7
Intoduction to Linux

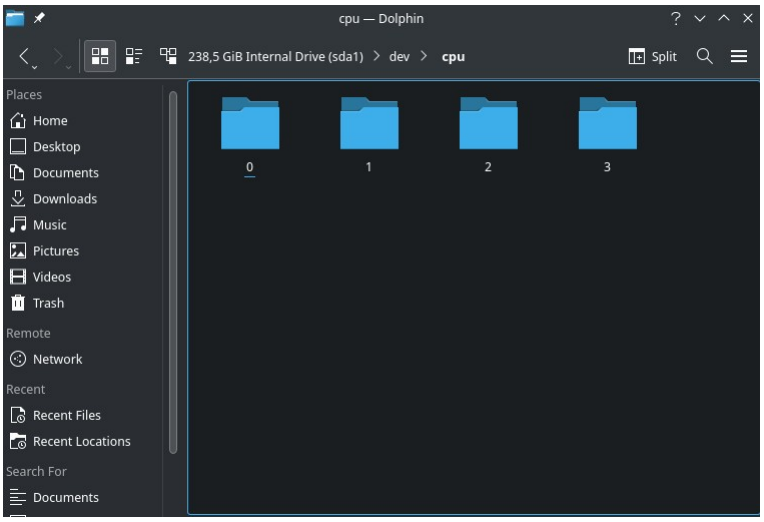
ჩვენ დაგეხმარებით ლინუქსის ინსტალაციაში.რატომ ლინუქსი?ლინუქსი ბევრად მოქნილი და გახსნილია ვიდრე სხვა ოპერატიული სისტემები.თანაც

ლინუქსი ბევრად ლაითი და მოსახერხებელია.ჩვენ ვფიქრობთ, რომ დამწყებთათვის იდეალური ლინუქს დისტროებია: Debian, Ubuntu, Manjaro და Mint.აქ მოცემულია ყველა დისტროს home საიტი, საიდანაც შეძლებთ საინსტალაციო .iso ფაილის გადმოწერას:
Debian: https://www.debian.org/download
Ubuntu: https://ubuntu.com/#download
Manjaro: https://manjaro.org/download/
Mint: https://www.linuxmint.com/download.php

თქვენ დაგჭირდებათ USB ფლეშ დრაივი(ფლეშკა და უმჯობესია თუ იქნება 8GB ზომაში), შემდეგ ფლეშკაზე ჩაწერთ გადმოწერილ ფაილს და დააინსტალირებთ.თუ თქვენ Windowsის მომხმარებელი ხართ გადმოწერეთ პროგრამა rufus.თუ თქვენ Linux მომხმარებელი ხართ მაშინ საერთოდ ნუ გადმოიწერთ ზევით ჩამოთვლილ დისტროებს, რადგან არ გჭირდებათ(ისედაც ლინუქსზე ხართ).როდესაც Windowsზე rufus ჩამოტვირთავთ გახსენით პროგრამა და შეაერთეთ ფლეშკა კომპიუტერში.შემდეგ rufusდან აირჩიეთ გადმოწერილი ფაილი .iso გაფართოებით და დააჭირეთ startს(ამოვარდება WARNING ფანჯარა, რომელიც გაგაფრთხილებთ რომ გაგრძელების შემთხვევაში ფლეშკიდან ყველაფერი წაიშლება).როდესაც ჩატვირთვა დასრულდება გამორთეთ rufus და გამოაერთეთ ფლეშკა კომპიუტერიდან.თუ თქვენ გინდათ, რომ იგივე კომპიუტერზე დააყენოთ ლინუქსი, დატოვეთ ფლეშკა და დაარესტარტეთ კომპიუტერი.თუ ლინუქსის საინსტალაციო არ გაეშვა და ისევ ვინდოუსი ჩაირთო, ისევ დაარესტარტეთ კომპიუტერი, შემდეგ გადადით BIOSში/UEFIში და მიანიჭეთ ფლეშკას high priority.ამის შემდეგ ისევ დაარესტარტეთ კომპიუტერი და ლინუქსის საინსტალაციო გაეშვება.

ლინუქსის ინსტალაცია გავიარეთ.ახლა საჭიროა, რომ გავიაზროთ თუ როგორ მუშაობს ლინუქსი(ანუ ლინუქსის ეკოსისტემას გავუგოთ).ლინუქსზე ყველაფერი ფაილებით აღიწერება.აი მაგალითად თუ თქვენს პროცესორს აქვს 4 ბირთვი და თქვენ დაწერთ ლინუქსის ტერმინალში ასეთ ბრძანებას:

თქვენ დაინახავთ, რომ ნაჩვენებია 4 ფოლდერი(folder), რომელთა დასახელებებია: 0, 1, 2, 3.თითო ფოლდერი აღნიშნავს პროცესორის თითო ბირთვს.ჩვენ რომ FM(File Manager) გავხსნათ და შევამოწმოთ ვალიდურია თუ არა ბრძანების output, ის გვაჩვენებს რომ ვალიდურია:



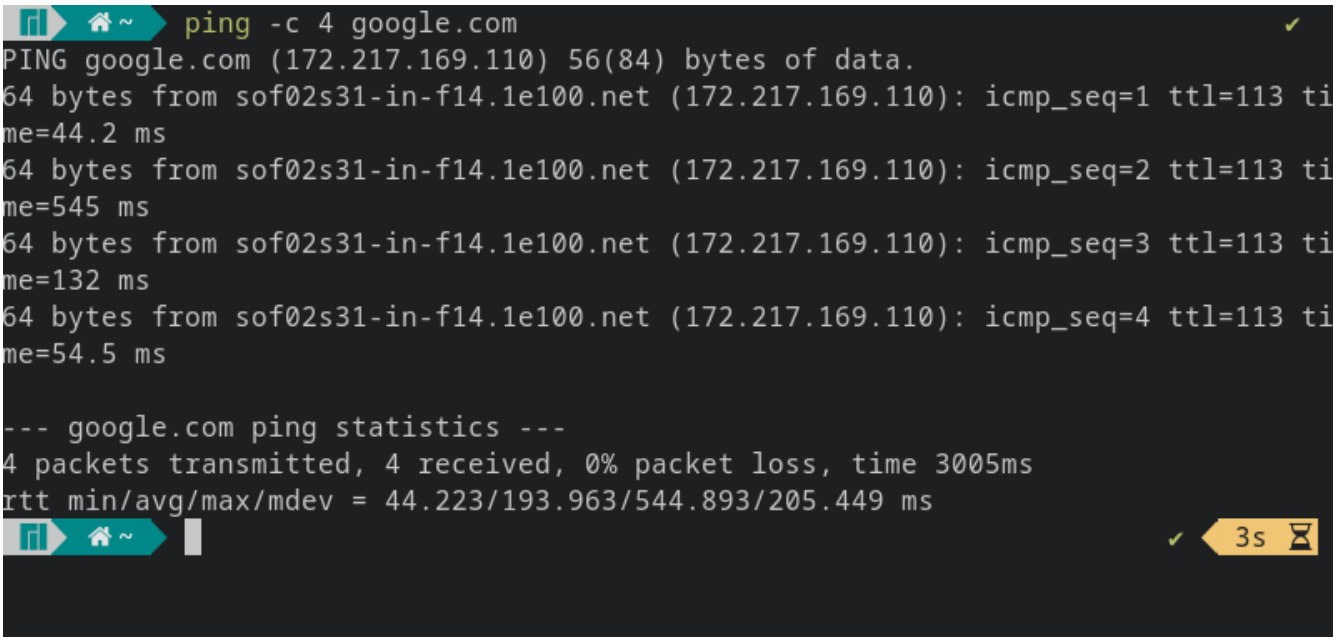
აქედან გამომდინარე ვასკვნით, რომ ლინუქსი ყველაფერს ფაილებს ამაგრებს.

თავი 7.1
Introduction to Linux Terminal

რა არის ტერმინალი(Terminal)?ტერმინალი არის პროგრამა, რომელიც საშუალებას იძლევა ტექსტური ბრძანებები გადავცეთ ოპერატიულ სისტემას.მაგალითისათვის ჩვენ შეგვიძლია დავწეროთ ტერმინალში:

```
ping -c 4 google.com
```

ამ ბრძანებას თუ გავუშვებთ, კომპიუტერი 4ჯერ დაპინგავს googleის სერვერს.ასეთი იქნება output:



ეს ნიშნავს, რომ პინგ მოთხოვნაზე გვიპასუხა სერვერმა(ასე ამოწმებენ ზოგჯერ არის თუ არა ინტერნეტი).ახლა განვიხილოთ ტერმინალი როგორ მუშაობს.ადამიანები როდესაც ტერმინალში მუშაობენ(ჰაკავენ, აპროგრამებენ, რაიმეს აკონფიგურირებენ, რაიმეს ასწორებენ), ისინი სხედან shellში.

* რა არის shell?

shell ქართულად ნიჟარას ნიშნავს.shell არის ინტერფეისი(Interface), რომელიც გვაძლევს საშუალებას კერნელს და ოპერატიულ სისტემას მარტივად და ადამიანისთვის გასაგებ ენაზე ვესაუბროთ.

shell ბევრნაირი არსებობს.მაგალითად: sh(shell), bash(bourne again shell), zsh(z shell), fish(friendly interactive shell).დიდი მნიშვნელობა არ აქვს რომელს გამოიყენებთ, უბრალოდ კომფორტის საკითხია.სქრინშოტებში რასაც ხედავთ ეს არის zsh v5.8.1 და საკმაოდ ლამაზიცაა.მაგრამ არის shellები, რომლებიც ძალიან ძველია და დიდად არც სილამაზით გამოირჩევა.ამ სქრინშოტში ხედავთ ცვლილებას zshდან shზე:

როგორც ხედავთ zshს უფრო მეტი თვისება აქვს:

1) უფრო ლამაზი promptის დიზაინი აქვს(მწვანე და თეთრი)

2) თუ ბრძანება არასწორია დაწერილ ბრძანებას გააწითლებს, ხოლო თუ სწორია მწვანედ დატოვებს.
3) აქვს auto-complition(ანუ თუ რაღაც ბრძანება ადრე დაწერილი გაქვთ და დაახლოებით იგივეს წერას დაიწყებთ, zsh ნაცრისფრად დაგიწერთ წინ ტექსტს თუ რა გეწერათ წინაზე(კარგია მაშინ, როდესაც დიდი ბრძანებაა და გავიწყდებათ).

shს კი დიდად არაფერი გააჩნია.არც promptის დიზაინია კარგი და არც კარგი ფუნქციები აქვს.

თავი 7.2
Introduction to Linux Filesystem

ლინუქსის ფაილების მენეჯმენტი ერთ-ერთი უმთავრესი რამაა.ქვემოთ სქრინშოტში ნაჩვენებია ყველა ის ფოლდერი და ფაილი, რაც მთავარ დირექტორიაში(ფოლდერში) ყრია(ანუ პირდაპირ დისკზე):

ფერებსაც დააკვირდით.ფერები უმნიშვნელოვანესია.ისინი გვეხმარება გავარჩიოთ თუ რომელი ფოლდერია(folder) და რომელი ფაილი(file).ლურჯი ტექსტი და ასევე tmp(მწვანე backgroundით რომ არის) არის ფოლდერები, დანარჩენი ყველა ფაილებია.ახლა ყველას სათითაოდ ავხსნით:

1) /bin – ამ ფოლდერში ყრია executables(ანუ მარტივად რომ ვთქვათ ბრძანებები და ისეთი ფაილები, რომელთა გაშვებაც შესაძლებელია).აი მაგალითად ბრძანება “ls” როდესაც ვწერ, ტერმინალი მიდის /bin ფოლდერში და ეძებს ფაილს სახელად “ls”.თუ იპოვის ის მას გაუშვებს და outputს ტერმინალში გამოიტანს, ხოლო თუ ვერ იპოვის დაწერს, რომ ფაილი ვერ იპოვა.
2) /dev – ამ ფოლდერში ყველა ის მოწყობილობა(device) არის მოცემული, რომელიც კომპიუტერზეა მიერთებული ან კომპიუტერის კომპონენტია(ზევით ხომ განვიხილეთ პროცესორის ბირთვები).
3) /lib – ამ ფოლდერში ყრია ბიბლიოთეკები(library).ბიბლიოთეკები პროგრამირებაში გამოიყენება.მათში სპეციალური კოდებია მოთავსებული, რომლებსაც საკმაოდ საინტერესო ფუნქციები

აქვს.მაგალითად Pythonს აქვს ბიბლიოთეკა სახელად socket.ეს ბიბლიოთეკა საშუალებას გვაძლევს Pythonდან ქსელში ვიმოქმედოთ(დავუკავშირდეთ პორტებს, გავხსნათ პორტები, მონაცემები გავაგზავნოთ სადმე ან მივიღოთ და ა.შ.)
4) /mnt – ეს ფოლდერი საკმაოდ საინტერესოა.აქ ნაჩვენებია ყველა storage სახის კომპონენტი რაც კომპიუტერზეა მიერთებული(HDD, SSD, USB, CD).
5) /root – ეს ფოლდერი არის root მომხმარებლის(ადმინისტრატორის) home ფოლდერი.
6) /sbin – ესეც იგივეა, რაც /bin მაგრამ აქ ყრია ადმინისტრატორისათვის განკუთვნილი executables და მათზე ხელი მხოლოდ root მომხმარებელს მიუწვდება.
7) /tmp – R00tM3 მას ხუმრობით RAMადაც მოიხსენიებს.ამ ფოლდერში ინახება ყველა ის მონაცემი, რასაც გაშვებული პროგრამები(მაგალითად firefox) მოიხმარს.
8) /boot – ამ ფოლდერში მოთავსებულია boot loader(OSის გამშვები) და ყველა ის საჭირო ფაილი და ფოლდერი, რომელიც მონაწილეობას იღებს OSის გაშვებაში.
9) /etc – ამ ფოლდერში მოთავსებულია სხვადასხვა .conf ან .config ფაილები(ანუ ისეთი ფაილები, რომლებიც მოწყობილობის ან პროგრამების კონფიგურაციაში იღებს მონაწილეობას).
10) /lib64 – ამ ფოლდერში 64 ბიტიანი ბიბლიოთეკებია მოთავსებული.
11) /opt – ამ ფოლდერში არის option პექიჯები.
12) /srv – აქ არის ისეთი სკრიპტები, რომლებიც სჭირდებათ მაგალითად ვებ სერვერებს.
13) /usr – ეს ფოლდერი შეიცავს MultiUser აპლიკაციებს.
14) /home – ეს არის home ფოლდერი ყველა ისეთი მომხმარებლისთვის, რომელიც არ არის root მომხმარებელი(ადმინისტრატორი).
15) /proc – ამ ფოლდერში არის ის პროცესები, რომლებსაც პროცესორი ასრულებს.
16) /run – აქ არის ინფორმაცია ბოლო bootდან.
17) /sys – ეს ფოლდერი შეიცავს ინფორმაციას მოწყობილობებზე, დრაივერებზე და კერნელზე.
18) /var – აქ არის ისეთი ფაილები, რომელთა მნიშვნელობებიც იცვლება. მაგალითად log ფაილები.

თავი 7.3
Introduction to Linux Package Managers

PMები(Package Manager) ლინუქსის განუყრელი მეგობრები არიან.პექიჯ მენეჯერები არის გზა, რომ რაიმე software დავაყენოთ ლინუქსზე.მაგალითად თუ თქვენ ხართ Windowsის მომხმარებელი, თქვენ სტიმს(steam) იწერთ სტიმის

ვებსაიტიდან.შემდეგ .exe ფაილს გახსნით დააწვებით რამდენჯერმე “Next”ს და შემდეგ “Done”.ამის შემდეგ უკვე სტიმს რთავთ.ლინუქსის შემთხვევაში საქმე გაცილებით ადვილადაა.ყველა დისტროს თავისი PM აქვს.მაგალითად Debianზე დაფუძნებული ყველა დისტროს PM არის “apt”.მაგალითად archზე დაფუძნებული ყველა დისტროს PM არის “pacman”.მაგალითად Voidის PM არის (xbps).მაგალითად თქვენ, რომ ანდროიდზე termux(ტერმინალი) გადმოიწეროთ თქვენ გექნებათ ორი PM.პირველი იქნება “apt” და მეორე “pkg”. მათი გამოყენება ძალიან ადვილია.დავუშვათ ჩვენ ვიყენებთ Ubuntu ლინუქსს(Debianზე დაფუძნებულია).ჩვენ გავხსნიდით ტერმინალს და ჩავწერდით:

sudo apt install steam -y

და მზადაა.ის გადმოიწერს სტიმს და დააინსტალირებს.საერთოდ არ გჭირდებათ ჯერ რაიმე პროგრამის ვებსაიტზე გადასვლა, შემდეგ მისი გადმოწერა, გახსნა, შემდეგ ღილაკებზე ჭერა და ბოლოს დასრულება.აქ ერთი პატარა ბრძანებით დავაყენეთ სტიმი.მაგალითად ეს რომ გვექნა arch ლინუქსზე ასეთი იქნებოდა ბრძანება:

sudo pacman -Sy steam

ხოლო void ლინუქსზე ასეთი იქნებოდა:

sudo xbps-install steam -y

ასეთი გამოსადეგია Package managerები.თქვენ თუ სისტემის განახლება მოგინდებათ არც მაგაზეა პრობლემა.აი სხვადასხვა Pmებზე როგორ უნდა განაახლოთ სისტემა:

```
sudo apt update -y
sudo pacman -Syu
sudo xbps-install -Syu
```

თქვენ რომ Windowsის მომხმარებელი იყოთ, დარწმუნებული ვარ დაიტანჯებოდით.ჩვენ ვფიქრობთ, რომ ძალა სიმარტივეშია და ეს ლოგიკურიცაა.Windowsის მომხარებელს რამდენი უნდა ეწვალა ეს ორი რამ რომ გაეკეთებინა, ლინუქსის მომხმარებელმა კი ეს უბრალოდ 2 პატარა ბრძანებაში შეძლო.აი თუ ლინუქსის მომხმარებელს ორივეს ცალ-ცალკე კეთება დაეზარა შეუძლია, რომ ეს ორი ბრძანება გააერთიანოს და თვითონ სხვა რამ გააკეთოს.აი ასეთი იქნებოდა გაერთიანებული ბრძანება:

sudo pacman -S steam && sudo pacman -Syu

ეს arch ლინუქსზე, მაგრამ სხვა ყველა დისტროზეც ანალოგიურად ხდება.უბრალოდ სხვადასხვა დისტროები სხვადასხვა PMებს ხმარობენ.

თავი 7.4

Introduction Linux to Services

ალბათ სერვისები ერთ-ერთი ყველაზე მნიშვნელოვანი თემაა, როდესაც ლინუქსთან გვაქვს საქმე.სერვისი შეიძლება იყოს gdm(DE), apache2(Web server), sshd(SSH server), vsftpd(FTP server) და კიდევ სხვა მრავალი.საქმე ისაა, რომ სხვადასხვა დისტროებზე სერვისებს სხვადასხვაგვარად ვრთავთ. მაგალითად ავიღოთ Debianზე დაფუძლებული დისტროები:

sudo service apache2 start

ასეთი ბრძანების დაწერა მოგვიწევდა იმისათვის, რომ HTTP სერვერი ჩაგვერთო.ახლა ავიღოთ archზე დაფუძნებული დისტოები:

sudo systemctl enable apache

ჩვენი აზრით ყველაზე კარგი ვარიანტი voidმა შემოგვთავაზა.სერვისებს კი არ ვრთავთ, არამედ ვყრით ფაილში, რომლიდანაც ისინი ავტომატურად ირთვება.void ლინუქსი ასე წყვეტს ამ პრობლემას.აიღეს /etc ფოლდერი, შიგნით ჩადეს sv ფოლდერი და ამ sv ფოლდერში ყრია ყველა შესაძლო სერვისი რისი ჩართვაც შეგვიძლია.შემდეგ ამ sv ფოლდერიდან ვლინკავთ ამ სერვისს /var ფოლდერში მდებარე /service ფოლდერში.ანუ ასეთი გამოდის საბოლოო ბრძანება:

sudo sv /etc/sv/anyServiceYouWant /var/service

არის მომენტები, როდესაც სერვისებს ვრთავთ, მაგრამ თვითონ არ ირთვებიან.ასეთ დროს უმჯობესია log ფაილებს გადახედოთ და errorის solution ინტერნეტში იპოვოთ.

თავი 8
Python

პითონი ერთ-ერთი ყველაზე პოპულარული პროგრამირების ენაა დღესდღეობით.მას უამრავ რამეში იყენებენ.მაგალითად: მონაცემთა ანალიზში, ვებ დეველოპმენტში(Back-End ფრეიმვორკებია: Django, Flask), თამაშების შექმნაში და ასე შემდეგ.ჰაკერებიც საკმაოდ ხშირად იყენებენ პითონს, კონკრეტულად scapy ბიბლიოთეკას(პაკეტების კრაფტია შესაძლებელი და მაგიტომ).ჩვენ გასწავლით პითონის საბაზისო სინტაქსს(რომელიც სხვათაშორის უფასოა, მაგრამ საქართველოში აკადემიები ამის სწავლებაში ფულს ახდევინებენ...არადა 25 წუთის საქმე ძლივსაა) და რამდენიმე ბიბლიოთეკასაც შევეხებით, კონკრეტულად: socket, threading და scapy.

თავიდან უნდა გაიგოთ ის, რომ პითონი არის ინტერპრეტირებული პროგრამირების ენა, ანუ ის არ კომპილირდება.ის არის High level

პროგრამირების ენა(ანუ დიდას სწრაფი ვერაა, მაგრამ კარგი ენაა).სხვათაშორის დამწყებთათვის ერთ-ერთი იდეალურია.

პითონის ინსტალაცია: თუ თქვენ ლინუქსის მომხმარებელი ხართ, თქვენს PMს მიუთითეთ, რომ დააინსტალიროს python და python3-pip(პითონის PM), ხოლო თუ თქვენ Windowsის მომხმარებელი ხართ, გადადით პითონის საიტზე და გადმოწერეთ ისა(შემდეგ დააინსტალირეთ).

ქვეთავი: ცვლადები

პროგრამირებაში ყოველთვის იყენებენ ცვლადებს.მაგრამ რა არის ცვლადი?ცვლადი შეგვიძლია წარმოვიდგინოთ, როგორც ყუთი სადაც შეგვიძლია რაიმე მონაცემი შევინახოთ.მაგალითად:

```
x = 10
```

ეს ნიშნავს იმას, რომ გვაქვს ცვლადი x, რომლის მნიშვნელობაც არის 10.ჩვენ შეგვიძლია ნებისმიერი მნიშვნელობა მივანიჭოთ მას.მაგალითად:

```
x = ‘Hello World’
```

ცვლადს ნებისმიერი სახელი შეგვიძლია დავარქვათ, მაგრამ სახელი არ უნდა იწყებოდეს რიცხვზე და სახელში ე.წ. “სფეისები” არ უნდა იყოს.

ქვეთავი: If else

ცხოვრებაში არის მომენტები, როდესაც არჩევანის გაკეთება გვიწევს.პროცესორის ცხოვრებაშიც დგება მომენტები, როდესაც ის არჩევანს აკეთებს.აი მაგალითად განვიხილოთ კოდი:

```
x = 10

if x > 5:
	print(“x > 5”)
else:
	print(“x < 5”)
```

როგორც ხედავთ, თავიდან x გავუტოლეთ 10ს.შემდეგ მოდის statement თუ x მეტია ხუთზე, მაშინ ვიძახებთ ფუნქციას print() და მას ეკრანზე გამოაქვს ტექსტი x > 5 თუ x 5ზე მეტი არაა მაშინ ისევ ვიძახებთ ფუნქციას print() და ამჯერად მას გამოაქვს ეკრანზე ტექსტი x < 5.

ასევე ჩვენ შეგვიძლია კოდში რამდენიმე statement წამოვწიოთ.მაგალითად განვიხილოთ კოდი:

```
x = 10

if x > 5:
	print(“x > 5”)
```

```
elif x == 5:
        print(“x = 5”)
else
        print(“x < 5”)
```

თავიდან x გავუტოლეთ 10ს.შემდეგ დავიწყეთ შედარება ანუ თუ x მეტია 5ზე ეკრანზე ვბეჭდავთ ტექსტს x > 5 თუ x 5ზე მეტი არაა, მაშინ გადავდივართ შემდეგ statementზე.ახლა ვადარებთ თუ x არის 5ის ტოლი(იქ ორი ტოლობა იმიტომ წერია, რომ პროგრამირებაში ერთი ტოლობა ნიშნავს ცვლადის რაიმესთვის გატოლებას, აქედან გამომდინარე ჩვენ, რომ იქ ერთი ტოლობა დაგვეწერა, ინტერპრეტერი იფიქრებდა რომ ჩვენ x 5ს გავუტოლეთ, არადა ჩვენ უბრალოდ მათ ერთმანეთს ვადარებთ.ამიტომ მოიგონეს ორი ტოლობა, რათა ერთმანეთს შეადარონ ცვლადი და მეორე ცვლადი ან ცვლადი და რაიმე მონაცემი) მაშინ ეკრანზე ვბეჭდავთ ტექსტს x = 5.ხოლო თუ ზემოთ მოცემული ორი პირობიდან არც ერთია ვალიდური, მაშინ ეკრანზე ვბეჭდავთ ტექსტს x < 5.

ქვეთავი: while

კოდის წერის დროს ციკლები უმნიშვნელოვანესია.რატომ?იმიტომ, რომ ისინი უამრავ პრობლემას ჭრიან.მათი დაწერა ძალიან ადვილია.მაგალითად განვიხილოთ კოდი:

```
x = 1

while x <= 1000:
        print(x)

        x = x + 1
```

თავიდან x გავუტოლეთ 1ს.შემდეგ ვწერთ სანამ x ნაკლებია ან ტოლი 1000ზე(ანუ არამკაცრი უტოლობა) დაბეჭდოს ეკრანზე xის მნიშვნელობა, შემდეგ კი x გაზარდოს ერთით.თუ ჩვენ ამ პროგრამას გავუშვებთ, ეკრანზე დაიბეჭდება რიცხვები 1დან 1000ის ჩათვლით.

ქვეთავი: ფუნქციები

ფუნქციები ყველაზე მეტად მგონი D1sM3ს უყვარს.პროგრამირებაში ფუნქციები ზუსტად იგივე როლს თამაშობს, რა როლსაც მათემატიკაში ასრულებს.ფუნქციაში ვწერთ რაიმე კოდს და ამ ფუნქციის გამოძახების შემდეგ ეს კოდი სრულდება.შესაძლოა ფუნქცია იყოს არგუმენტიანი და მრავალარგუმენტიანი.ან შესაძლოა ფუნქციას საერთოდ არ ჰქონდეს არგუმენტი.არგუმენტებს მოგვიანებით განვიხილავთ.ასე იწერება ფუნქცია პითონში:

```
def MyFunc():
        print(“Hello World!”)
```

```
MyFunc()
```

როგორც ხედავთ თავიდან ვწერთ “def” სიტყვას.ეს ნიშნავს, რომ ჩვენ ვქმნით ახალ ფუნქციას.შემდეგ იწერება ფუნქციის სახელი(რაც გინდათ ის დაარქვით.ამ შემთხვევაში დავარქვით MyFunc).ფუნქციის სახელის გვერდით რომ ორი ფრჩხილია მაგათ შორის იწერება არგუმენტები(არაა სავალდებულო). შემდეგ ფუნქციაში ვიძახებთ print() ფუნქციას, რომელიც ეკრანზე დაბეჭდავს ტექსტს Hello World! და შემდეგ ჩვენი დაწერილი კოდი გაითიშება.სულ ბოლოს კი ჩვენს მიერ შექმნილ ფუნქციას ვიძახებთ.ამ კოდის გაშვების შემდეგ, ეკრანზე დაიბეჭდება ტექსტი Hello World! და შემდეგ პროგრამა გამოირთვება.

არგუმენტიანი ფუნქციები საკმაოდ ადვილი გასაგებია.არგუმენტი არის ნებისმიერი სახის მონაცემი.ჩვენ ფუნქციას ვაწვდით არგუმენტს, ის კი ამ არგუმენტს რაღაცაში(რასაც თქვენ ეტყვით იმაში) იყენებს.მაგალითად განვიხილოთ კოდი:

```
def MyFunc(x):
    print(x)

MyFunc(“Hello World!”)
```

თავიდან ვქმნით ფუნქციას სახელად “MyFunc” და ვაძლევთ არგუმენტს სახელად x.შემდეგ ვიძახებთ ფუნქციას და შიგნით ვწერთ ტექსტს Hello World! რის შემდეგაც თუ ამ კოდს ჩავრთავთ ის ეკრანზე დაბეჭდავს ტექსტს Hello World!

მრავალარგუმენტიანი ფუნქციები კი ზუსტად იგივეა რაც არგუმენტიანი ფუნქციები, უბრალოდ მათ მეტი არგუმენტი აქვთ ვიდრე ერთარგუმენტიანს.განვიხილოთ კოდი:

```
def MyFunc(x, y):
    print(x + y)

MyFunc(10, 20)
```

თავიდან შევქმენით ფუნქცია სახელად “MyFunc” და გადავაწოდეთ ორი არგუმენტი x და y.შემდეგ გამოვიძახეთ ეს ფუნქცია და მივეცით რიცხვები 10 და 20.ფუნქციაში წერია, რომ მან xსა და yის ჯამი უნდა დაწეროს ეკრანზე.ანუ ის დაწერს ეკრანზე 30ს, რადგან 10 + 20 = 30.

ქვეთავი: ბიბლიოთეკები

როგორც ყველა პროგრამირების ენას, ისე პითონსაც აქვს ბიბლიოთეკები.ბიბლიოთეკები არის დაწერილი კოდები, რომლებშიც არის სპეციალური ფუნქციები.პითონს აქვს ჩაშენებული ფუნქციები, მაგალითად print() მაგრამ ყველა ფუნქციის ჩაშენება შეუძლებელია, ასე რომ მოიგონეს ბიბლიოთეკები.განვიხილოთ კოდი:

```
import socket

socket.connect(“www.google.com”, 443)
```

თავიდან ჩვენ შემოვიტანეთ socket ბიბლიოთეკა, შემდეგ კი socket ბიბლიოთეკიდან გამოვიძახეთთ ფუნქცია connect(), რომელსაც გადავეცით ორი არგუმენტი, პირველი ვებსაიტის ლინკი, ხოლო მეორე მისი პორტი(HTTPS).ამ კოდის გაშვების შემდეგ, კომპიუტერი დაუკავშირდება googleის სერვერის HTTPS პორტს(რადგან HTTPS პორტი არის 443 პორტი).

ქვეთავი: პითონის PM(Package Manager)

პითონსაც აქვს თავისი PM.მას ეწოდება pip.მისი დახმარებით შეგვიძლია პითონის ბიბლიოთეკები გადმოვიწეროთ.მაგალითად ჩვენ გვინდა, რომ გადმოვწეროთ scapy.ჩვენ ვწერთ ბრძანებას:

```
sudo pip3 install scapy
```

ამის შემდეგ შეგვიძლია ის დავაიმპორტოთ კოდში, ან პირდაპირ გავხსნათ ის(ტერმინალიდან).

ქვეთავი: Arrays

ეს ორი საკმაოდ საინტერესოა.მაგალითად გვინდა, რომ რამდენიმე მნიშვნელობა გვქონდეს ერთ ცვლადში.ვქმნით სიას და მასში ვამატებთ რამდენიმე მნიშვნელობას:

```
BookAuthors = [“61c4da”, “<3 D1sM3 <3”, “N3tR3t”, “R00tM3”]
```

და თუ გვინდა, რომ კოდიდან დავამატოთ რაიმე ახალი მნიშვნელობა სიაში ვაკეთებთ ამას:

```
myArray = []

myArray.append(“something here”)
```

ქვეთავი: try && except

ეს ნამდვილად გამოგადგებათ.მაგალითად გინდათ, რომ რაიმე სცადოთ მაგრამ თუ ეს არ იმუშავებს არ გინდათ outputად რაღაც საშინელი error.აკეთებთ ასეთ რამეს:

```
try:
	print(x)
except:
	print(“ვერ ვიპოვე x”)
```

ამ პროგრამას თუ გაუშვებთ ეკრანზე დაიბეჭდება ტექსტი:
ვერ ვიპოვე x

რადგან x ცვლადი არ არსებობს კოდში.

ქვეთავი: user input

თუ გინდათ, რომ კოდში მომხმარებელმა რაიმე შეიყვანოს მაშინ გამოიყენეთ პითონში ჩაშენებული ფუნქცია input() აი ასე:

password = input(“Enter password: “)

განვიხილოთ კოდი.password არის ცვლადი, რომელიც გავუტოლეთ input() ფუნქციას.ანუ ამ კოდს როდესაც გავუშვებთ, ჯერ ეკრანზე დაიწერება ტექსტი:
Enter password:

და შემდეგ, როდესაც მომხმარებელი შეიყვანს ტექსტს და “Enter”ს დააწვება პროგრამა გაითიშება(password იმას გაუტოლდება, რაც მომხმარებელმა დაწერა).

ქვეთავი: კომენტარები

ყველა პროგრამირების ენას აქვს კომენტარი.კომენტარი იმისათვის გამოიყენება, რომ უკეთ გავიგოთ კოდი რას აკეთებს.კომენტარი იწერება “#” სიმბოლოს შემდეგ:

```
# აქ ეკრანზე დავბეჭდეთ ტექსტი “Hello World!”
print(“Hello World!”)
```

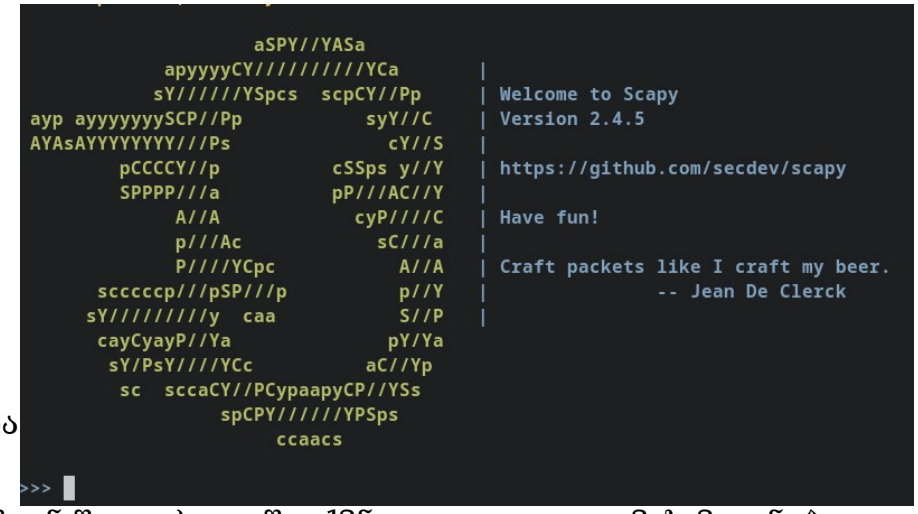
ამ პროგრამის გაშვების შემდეგ ეკრანზე დაიბეჭდება ტექსტი Hello World!

მგონი პითონის very basicები აგიხსენით.ლოგიკას უკვე თქვენთვითონ ჩაწვდით.

თავი 9
პაკეტები

რა არის პაკეტები?პაკეტები არის მესამე დონეზე(მესამე თავში განვიხილეთ) მყოფი მონაცემები(ანუ ის data რაც იგზავნება, მაგალითად თამაშის კლიენტი სერვერს თავის კოორდინატებს უგზავნის).პაკეტები შეიცავენ რამდენიმე სახის ინფორმაციას.ახლა ჩვენ ჩავრთოთ scapy(პითონის ბიბლიოთეკა) და შევქმნათ პაკეტი:

ასე გამოიყურება scapy



ჩართვისას.ჩვენ შეგვიძლია შევქმნათ ცვლადი და მას მივანიჭოთ გარკვეული მნიშვნელობა:

```
>>> packet = IP()/TCP()
>>> packet
<IP  frag=0 proto=tcp |<TCP  |>>
>>>
```

როგორც ხედავთ, ჩვენ ცვლადს დავარქვით “packet” და მივანიჭეთ მნიშვნელობა “IP()/TCP()”

* რას ნიშნავს “IP()/TCP()”?
    scapy ბიბლიოთეკაში რა თქმა უნდა გვაქვს გარკვეული ფუნქციები და IP() და TCP() ფუნქციებიც მათ შორისაა.IP() ფუნქცია ნიშნავს იმას, რომ მონაცემები უნდა გაიგზავნოს მესამე დონეზე(layer 3) და TCP() ფუნქცია ნიშნავს, რომ პროტოკოლი უნდა იყოს TCP, ხოლო მათ შორის “/” ნიშანი უბრალოდ ამ ფუნქციებს ერთმანეთისგან გამოყობს(სინტაქსია ასეთი).

შემდეგ ჩვენ დავწერეთ “packet”, რათა გვენახა თუ რა მნიშვნელობის იყო იგი და ეკრანზე დაიბეჭდა პაკეტის სრული მნიშვნელობა.როგორც უკვე ვთქვით ქსელში არის მონაცემების წყარო და დანიშნულების ადგილი.ჩვენ პაკეტში ვწერთ ორივეს(წყაროს იმიტომ, რომ დანიშნულების ადგილმა გვიპასუხოს ჩვენ და არა სხვას, ხოლო დანიშნულების ადგილს იმიტომ ვწერთ, რომ პაკეტი მასთან მივიდეს):

```
>>> packet.dst = "192.168.1.1"
>>> packet.src = "192.168.1.6"
>>> packet
<IP  frag=0 proto=tcp src=192.168.1.6 dst=192.168.1.1 |<TCP  |>>
>>>
```

ჩვენ თავიდან პაკეტში ჩავწერეთ დანიშნულების წერტილის IPv4 მისამართი(ჩემი როუტერის IPv4 მისამართი), ხოლო შემდეგ ჩავწერეთ წყაროს

IPv4 მისამართი(ჩემი ტელეფონის, რადგან ტელეფონი როუტერზე მაქვს დაკავშირებული და კომპიუტერს ტელეფონიდან ვუზიარებ ინტერნეტს ანუ ტელეფონსა და ჩემს კომპიუტერს შორის კიდევ ერთი LAN არის 192.168.42.0/24 ქსელი).ჩვენ უკვე შეგვიძლია ეს პაკეტი გავაგზავნოთ, მაგრამ ასევე შეგვიძლია პაკეტში ჩავწეროთ მისი TTL(Time To Live).TTLის მაქსიმალური მნიშვნელობა არის 255, მაგრამ ვიკიპედია გვეუბნება რომ რეკომენდირებულია 64(R00tM3 სწორი აღმოჩნდა...).TTLის ჩაწერა ასე შეგვიძლია:

```
>>> packet.ttl = 10
>>> packet
<IP  frag=0 ttl=10 proto=tcp src=192.168.1.6 dst=192.168.1.1 |<TCP  |>>
>>>
```

როგორც ხედავთ, პაკეტში უკვეა მოცემული რამდენიმე ჰედერი(Header): წყაროს IPv4, დანიშნულების IPv4 და პაკეტის TTL.ჩვენ შეგვიძლია გამოვიყენოთ scapy ბიბლიოთეკაში ჩაშენებული send() ფუნქცია, რათა ეს პაკეტი გავაგზავნოთ.საჭირო არ არის send() ფუნქციისათვის დანიშნულების IPv4 მისმართის მითითება, რადგან ეს პაკეტში უკვე წერია:

```
>>> send(packet)
.
Sent 1 packets.
>>>
```

. ნიშნავს თუ რამდენჯერ გაიგზავნა პაკეტი.ჩვენ შეგვიძლია ციკლის დახმარებით ეს პაკეტი რამდენჯერაც გვინდა იმდენჯერ გავაგზავნოთ.მაგალითად 10ჯერ.ამისათვის საჭიროა უკვე ფაილში კოდის ჩაწერა.თავიდან საჭიროა რომელიმე text editorით გავხსნათ ახალი ფაილი და მასში ჩავწეროთ კოდი:

```
/tmp nvim testPacket.py                                    ✔
```

შემდეგ ვწერთ ასეთ კოდს:

```python
# ვაიმპორტებთ scapy ბიბლიოთეკას
from scapy.all import *

# ვქმნით layer 3 && TCP პაკეტს
packet = IP()/TCP()

# ვამატებთ ჰედერებს პაკეტში
packet.src = "192.168.1.6" # წყაროს IPv4
packet.dst = "192.168.1.1" # დანიშნულების IPv4
packet.ttl = 10            # TTL

# ვქმნით ცვლადს ციკლისათვის
x = 1

while x <= 10:
    # ვაგზავნით პაკეტს
    send(packet)

    # x გაიზარდოს ერთით, რათა ციკლი სამუდამო არ იყოს
    x = x + 1
```

კოდის განხილვას აღარ დავიწყებთ, რადგან ჩვენ უკვე მივუწერეთ კომენტარები კოდის ყველა ნაწილს.სანამ პაკეტს გავუშვებთ, უნდა ვთქვათ ის, რომ ამ პროგრამის გაშვება უნდა მოხდეს root მომხმარებლის მიერ.ეს იმიტომ, რომ მხოლოდ root მომხმარებელს აქვს უფლება პაკეტები დაკრაფტოს(craft) და გააგზავნოს.ჩვენ ვფიქრობთ, რომ ეს უსაფრთხოების საკითხია.მაგალითად შესაძლებელია, რომ ჩვეულებრივმა მომხმარებელმა რაიმე ვირუსი ჩაიწეროს კომპიუტერში, რომელიც ეცდება კომპიუტერიდან ცუდი პაკეტები გაისროლოს ქსელში(მაგალითად DOS შეტევა სცადოს), აქედან გამომდინარე თუ ჩვენ ჩვეულებრივ მომხმარებელს ავუკრძალავთ ქსელში პაკეტების გასროლის უფლებას, ვირუსი უძლური იქნება.ამიტომაც ჩვენ თუ გვინდა, რომ ამ კოდმა იმუშაოს საჭიროა root მომხმარებლის უფლებები მოვიპოვოთ.ეს შესაძლებელია sudo ბრძანების დახმარებით.ჩვენ ვწერთ ტერმინალში:

sudo python3 testPacket.py

და კოდი იმუშავებს:

```
/tmp sudo python3 testPacket.py
[sudo] password for cicada:
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
/tmp 5s
```

როგორც ხედავთ, დაკრაფტული პაკეტი 10ჯერ გაიგზავნა.ჩვენ შეგვიძლია პაკეტს სპეციფიკური პორტებიც მივაწოდოთ(ანუ წყაროს პორტი საიდანაც უნდა გაიგზავნოს მონაცემები და დანიშნულების ადგილის პორტი სადაც უნდა შევიდეს მონაცემები).მაგალითად ახლა ჩემი როუტერის HTTP პორტზე(80/TCP) გავაგზავნით პაკეტს.მაგრამ ჯერ უნდა გავიგოთ არის თუ არა გახსნილი როუტერზე HTTP პორტი.ამისათვის დაგვეხმარება ერთ-ერთი ყველაზე ცნობილი ხელსაწყო “nmap”.ეს ხელსაწყო სხვა მოწყობილობების პორტებს ასკანირებს და გვეუბნება თუ რა პორტებია გახსნილი remote მოწყობილობაზე:

```
~ nmap 192.168.1.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-16 13:04 +04
Nmap scan report for 192.168.1.1
Host is up (0.021s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT     STATE    SERVICE
53/tcp   open     domain
80/tcp   open     http
443/tcp  open     https
5060/tcp filtered sip

Nmap done: 1 IP address (1 host up) scanned in 1.54 seconds
~
```

როგორც ხედავთ, ჩემს როუტერს აქვს 4 პორტი გახსნილი(ბოლო დაფილტერებულია და არ გვეგონა, რომ HTTPS პორტი ექნებოდა ჩემს

როუტერს D1sM3მ რატომღაც იცოდა).ჩვენ გვჭირდება HTTP პორტი.რადგან ვიცით, რომ პორტი ღიაა, უკვე შეგვიძლია scapyში დავამატოთ ეს ინფორმაცია:

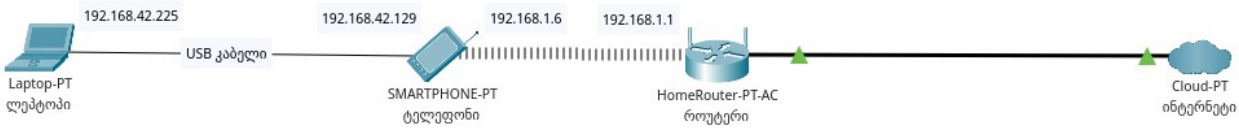
```
>>> packet = IP()/TCP()
>>> packet.src = "192.168.1.6"
>>> packet.dst = "192.168.1.1"
>>> packet.dport = 80
>>>
>>> send(packet)
.
Sent 1 packets.
>>>
```

როგორც ხედავთ, ჩვენ დავამატეთ წყაროსა და დანიშნულების მისამართები და ასევე დავამატეთ dport(Destination Port), ანუ დანიშნულების პორტი.შემდეგ გავაგზავნეთ პაკეტი.

* რა მოუვიდა პაკეტს?
    პაკეტი ჩემი კომპიუტერიდან გამოვიდა, შემდეგ გადავიდა ჩემს ტელეფონში(ზემოთ ვახსენე, რომ ტელეფონით ვაზიარებ კომპიუტერზე ინტერნეტს) და ტელეფონიდან მიუვიდა როუტერს(კონკრეტულად HTTP პორტს).

* მაგრამ თუ ტელეფონის IPv4 არის მითითებული წყაროში როუტერი კომპიუტერს არ დაუბრუნებს პასუხს?
    არა.როუტერი ფიქრობს, რომ პაკეტი ჩემი ტელეფონიდან გაიგზავნა, რადგან მისი IPv4 წერია პაკეტის ჰედერში.

აი მარტივად განახებთ რა მოხდა:



ასე გამოიყურება ახლა ქსელი.ახლა კი მოდით ცოტა ცუდ თემას შევეხოთ.დავუშვათ, ჩვენ გვინდა, რომ DOS შეტევა განვახორციელოთ სამიზნეზე.ეს როგორ უნდა გავაკეთოთ?საჭირო არაა დიდი სკრიპტების წერა.ჩვენ მხოლოდ პაკეტი გვჭირდება.საქმე ისაა, რომ ჩვენ უკვე გაჩვენეთ DOS შეტევა(ტექნიკურად).ანუ იმის თქმას ვცდილობთ, რომ რეალურად ტელეფონი საერთოდ არაფერ შუაშია ამ პაკეტებთან, ის უბრალოდ ინტერნეტს უზიარებს ჩემს ლეპტოპს(USB კაბელით).მაგრამ როდესაც მე პაკეტს ვაგზავნი, როუტერი ამ პაკეტს “192.168.1.6”ზე პასუხობს, ანუ პასუხი არა ჩემს კომპიუტერს, არამედ ტელეფონს მისდის(ტელეფონი არც კი უსმენს პასუხს, რადგან, როგორც უკვე ვთქვით, ის არაფერ შუაშია ამ დიალოგთან).აბა დაფიქრდით, თქვენი აზრით, როგორ შეიძლება ეს DOS შეტევა აღმოჩნდეს?

პასუხი ძალიან მარტივია.დავუშვათ ტელეფონს აქვს გახსნილი პორტი, მაგალითად SSH(Secure Shell).ჩვენ გვინდა, რომ ტელეფონზე ამ პორტით წვდომა ვერავინ აიღოს.ჩვენ შეგვიძლია ორნაირი სახის შეტევა განვახორციელოთ:

1) პირდაპირ ლეპტოპიდან “გავურაშოთ” ტელეფონის პორტს და flood შეტევა ვაწარმოოთ.
2) ჩვენ შეგვიძლია როუტერს გავუგზავნოთ პაკეტი სადაც ეწერება, რომ პაკეტი არის ტელეფონიდან წამოსული და პორტი კონკრეტულად არის 22(SSH).ამის შემდეგ როუტერი პასუხს გასცემს ტელეფონს 22-ე პორტზე.თუ ჩვენ ამას ლეპტოპიდან გაუჩერებლად გავაკეთებთ, საბოლოოდ როუტერი ტელეფონზე flood შეტევას აწარმოებს, თანაც ისე, რომ ვერაფერს მიხვდება.

აქედან გამომდინარე ჩვენ შეგვიძლია ასეთი კოდი დავწეროთ:

```
# ვაიმპორტებთ scapy ბიბლიოთეკას
from scapy.all import *

# ვქმნით ცვლადს packet და მას ვანიჭებთ მნიშვნელობას
packet = IP()/TCP()

# packetში ვწერთ ჰედერებს
packet.src = "192.168.1.6" # წყაროს IPv4 (ეს წყარო რეალურად არაა)
packet.dst = "192.168.1.1" # დანიშნულების IPv4
packet.sport = 22          # წყაროს პორტი(სადაც უნდა უპასუხოს როუტერმა)
packet.dport = 80          # დანიშნულების პორტი

# უსასრულო ციკლი, რათა გაუჩერებლად აგზავნოს დეპტოპლა პაკეტები როუტერთან
while True:
    send(packet)
```

კოდის განხილვას აღარ დავიწყებთ, ისევ აქვს კომენტარები დართული.ამ კოდის გაშვების შემდეგ(root უფლებებით უნდა გაუშვათ, თორემ არ იმუშავებს), როუტერი გაუჩერებლად მიიღებს პაკეტებს 80-ე პორტზე(HTTP) და როდესაც პაკეტს დაიკითხავს და ნახავს , რომ წყარო არის ჩემი ტელეფონი(არადა რეალურად არ არის ასე) და წყაროს პორტია 22-ე პორტი(SSH), ის ადგება და “წყაროს” უპასუხებს 22-ე პორტზე, ანუ ტელეფონს გაუგზავნის პაკეტებს 22-ე პორტზე.ამ ცუნამის(flood) შედეგად სხვა მომხმარებლები ვერ შეძლებენ ამ პორტის გამოყენებას, რადგან ის დაკავებულია მოთხოვნებზე(request) პასუხით.ამას ეწოდება “SYN Flood” შეტევა(attack).

თავი 10
Threading

რა არის წრედი(Thread)?წრედები საინტერესო საკითხია.წრედი არის გარკვეული Task(დავალება) პროცესორისათვის.პროგრამირებაში ძალიან მნიშვნელოვანია.მაგალითად გვაქვს პითონში შექმნილი ორი ფუნქცია.პირველი ფუნქცია მომხმარებლისგან იღებს ტექსტურ inputს და აგზავნის სერვერზე, ხოლო მეორე უკავშირდება სერვერს და იწყებს მისგან პასუხების მიღებას.ზოგადად პითონში და ყველა პროგრამირების ენაში ლოგიკა ერთია.ანუ ალგორითმული ლოგიკაც და მუშაობის პრინციპიც.მუშაობის პრინციპი ისაა, რომ კოდი ეშვება ზევიდან(კოდის პირველი ხაზიდან) და ჩამოდის ქვევით.ჩვენ შეგვიძლია ასეთი რამ დავწეროთ:

```
def f1():
    print(“f1”)

def f2():
    print(“f2”)

f1()
f2()
```

როგორც ხედავთ შევქმენით ორი ფუნქცია, რომლების გამოძახებისასაც ისინი თავიანთ სახელებს დაბეჭდავენ.ჩვენ ქვევით ორივე მათგანი გამოვიძახეთ, და როგორც ხედავთ, ჯერ შესრულდება f1 ფუნქცია, ხოლო შემდეგ f2 ფუნქცია.მაგრამ ზევით მოცემულ მაგალითში, ჩვენ არ გვაწყობს პასუხის მიღება და შემდეგ ტექსტის დაწერა და გაგზავნა.ჩვენ გვინდა, რომ ეს ორი რამ ერთდროულად მოხდეს.აქედან გამომდინარე გვჭირდება რაღაც მეთოდი, რომელიც დაგვეხმარება ამ პრობლემის გადაჭრაში.ეს მეთოდი გახლავთ წრედინგი(threading).პითონში არსებობს ბიბლიოთეკა, სახელად “threading”.ის გვეხმარება, რომ ორი ან მეტი ფუნქცია ერთდროულად ვამუშაოთ და არა ცალ-ცალკე, როგორც ეს ზემოთ მოცემულ კოდშია.ასეთი გამოვა წრედინგით გაკეთებული იგივე კოდი:

```python
# ვაიმპორტებთ threading ბიბლიოთეკას
import threading

# ვქმნით ფუნქციას f1
def f1():
    print("f1")

# ვქმნით ფუნქციას f2
def f2():
    print("f2")

# ვიწყებთ წრედების შექმნას(ვქმნით ცვლადებს threadf1 და threadf2)
# ქვედა ორ ხაზში ჩვენ ცვლადებს მივანიჭეთ threading ბიბლიოთეკაში მყოფი
# ფუნქციის Thread(ის მნიშვნელობა)
threadf1 = threading.Thread(target=f1)
threadf2 = threading.Thread(target=f2)

# ვრთავთ წრედებს
threadf1.start()
threadf2.start()
```

ეს კოდი იგივეს შეასრულებს, რაც ზემოთ მოცემულ კოდში სრულდებოდა, იმ განსხვავებით, რომ ტექსტები f1 და f2 ერთდროულად დაიბეჭდება და არა ცალ-ცალკე.შესაძლოა ჯერ ერთი ეწეროს და შემდეგ მეორე, მაგრამ საქმე ისაა, რომ ისინი ერთდროულად გახორციელდა.

თავი 11
sockets

სოკეტი(socket) არის პითონის ბიბლიოთეკა, რომელიც საშუალებას იძლევა ქსელში ორ მოწყობილობას შორის კავშირი დავამყაროთ.ჩვენ socketით შეგვიძლია ავაწყოთ სერვერები და კლიენტები.მაგალითად თქვენ თუ

გადახვალთ ამ ლინკზე, ნახავთ 6ic4da$(ანუ ჩემს მიერ) მიერ აწყობლ very simple ჩატს:

https://github.com/6icada/PyChat

სოკეტები შეგვიძლია ორნაირი კავშირისათვის გამოვიყენოთ.პირველია TCP, ხოლო მეორეა UDP(ესენი მესამე თავში განვიხილეთ).მაგალითად ავიღოთ კოდი:

```
import socket

HOST = “0.0.0.0”
PORT = 12
Server_Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Server_Socket.bind((HOST, PORT))
Server_Socket.listen()
```

შესაძლოა უცნაურად გამოიყურება, მაგრამ ყველაფერს აგიხსნით:

1) HOST და PORT არის ცვლადები, რომლებსაც აქვთ გარკვეული მნიშვნელობები.HOSTის მნიშვნელობაა ტექსტი 0.0.0.0 ანუ ეს იმას ნიშნავს, რომ როდესაც სერვერი ჩაირთვება, მან ყველა მხრიდან უნდა მოუსმინოს შემომავალ კავშირებს(ანუ თუ მოწყობილობას ერთზე მეტი IP მისამართი აქვს, ის ყველა მისამართზე მოუსმენს ამ პორტს).PORTის მნიშვნელობა კი არის რიცხვი 12 ანუ სერვერი მე-12 პორტზე მოუსმენს შემომავალ კავშირებს.

2) Server_Socket არის ცვლადი, რომლის მნიშვნელობა ოდნავ გრძელია და მათ ამ ნომრის ქვენომრებში განვიხილავთ.

2.1) socket.socket ნიშნავს, რომ ჩვენ socket ბიბლიოთეკიდან ვიძახებთ socket() ფუნქციას და მას გადავცემთ არგუმენტებს.

2.2) socket.AF_INET ეს არის socket.socket()ის პირველი არგუმენტი.რეალურად socket.AF_INET ნიშნავს, რომ socket ბიბლიოთეკიდან მოგვაქვს ცვლადი AF_INET, რაც ნიშნავს IPv4 მისამართს.

2.3) socket.SOCK_STREAM არის ცვლადი, socket ბიბლიოთეკიდან გადმოტანილი, ანუ ჩვენ გადმოგვაქვს socket ბიბლიოთეკიდან SOCK_STREAM ცვლადი, რაც ნიშნავს TCP კავშირს.

3) Server_Socket.bind((HOST, PORT)) რომ განვმარტოთ ასეთ რამეს მივიღებთ: Server_Socket უკვე გატოლებულია(კოდის ზედა ხაზში) socket.socket() ფუნქციასთან, რომელსაც გადავეცით ორი არგუმენტი(socket.AF_INET და socket.SOCK_STREAM) და აქედან გამომდინარე აქედან უკვე შეგვიძლია სხვადასხვა ფუნქციები გამოვიყენოთ.მაგალითად აქ ვიყენებთ bind() ფუნქციას, რომელსაც გადაეცემა ე.წ tuple, რომელიც არის (HOST, PORT).ანუ ეს ხაზი სერვერს სვამს 0.0.0.0:12ზე(0.0.0.0 IPv4 არის და 12 პორტი).

4) Server_Socket.listen() ეს ნიშნავს, რომ წინა ხაზზე რომ დავსვით სერვერი 0.0.0.0:21ზე, ის სერვერი იწყებს მოსმენას ანუ იხსნება მე-12 პორტი.

ამ კოდის ქვემოთ უკვე შეგვიძლია დავამატოთ ფუნქციები, რომლებსაც გამოვიყენებთ სერვერისთვის, მაგალითად ჰენდლერ(handler) ფუნქცია, რომელიც მიიღებს კლიენტებს და მაგალითად მათ მოათავსებს რაიმე სიაში ან მესიჯსენდერი(message sender) ფუნქცია, რომელიც სიაში ყველა კლიენტს ჩამოუვლის და ყველას გაუგზავნის რაღაც მესიჯს.

* Tip: სანამ სოკეტებით სხვა მოწყობილობას რაიმე მესიჯს გაუგზავნით არ დაგავიწყდეთ ამ მესიჯის დაენკოდება(encode).დაენკოდება ნიშნავს მაგალითად რიცხვის ან რაიმე ტექსტის ბაიტებში გადაყვანას, რათა შემდეგ ის ქსელში გაიგზავნოს.როდესაც დააენკოდებთ და გააგზავნით მესიჯს და მიმღებთან მივა მესიჯი, არ დაგავიწყდეთ იმ მესიჯის დადეკოდება(decode).დადეკოდება ნიშნავს ბაიტებში გადაყვანილი მესიჯის ტექსტად ან რიცხვად გადაქცევას.

მაგალითისათვის ასეთი იქნება მარტივი ჰენდლერ სერვერი:

```
import socket

HOST = “0.0.0.0”
PORT = 12
Server_Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clients = []

Server_Socket.bind((HOST, PORT))
Server_Socket.listen()

def Handle():
	while true:
		client, address = Server_Socket.accept()

		clients.append(address)
Handle()
```

განვიხილოთ კოდი.თავიდან, რა თქმა უნდა, ვაიმპორტებთ socket ბიბლიოთეკას, შემდეგ მოდის ცვლადების შექმნა და სერვერის ჩართვა და აი ბოლოს ვქმნით ფუნქციას.მოდით დავაკვირდეთ და განვიხილოთ ეს ფუნქცია: ფუნქციაში არის უსასრულო ციკლი, რადგან მან უსასრულოდ მიიღოს კლიენტები.კლიენტების მიღება ხდება ამ ხაზით:

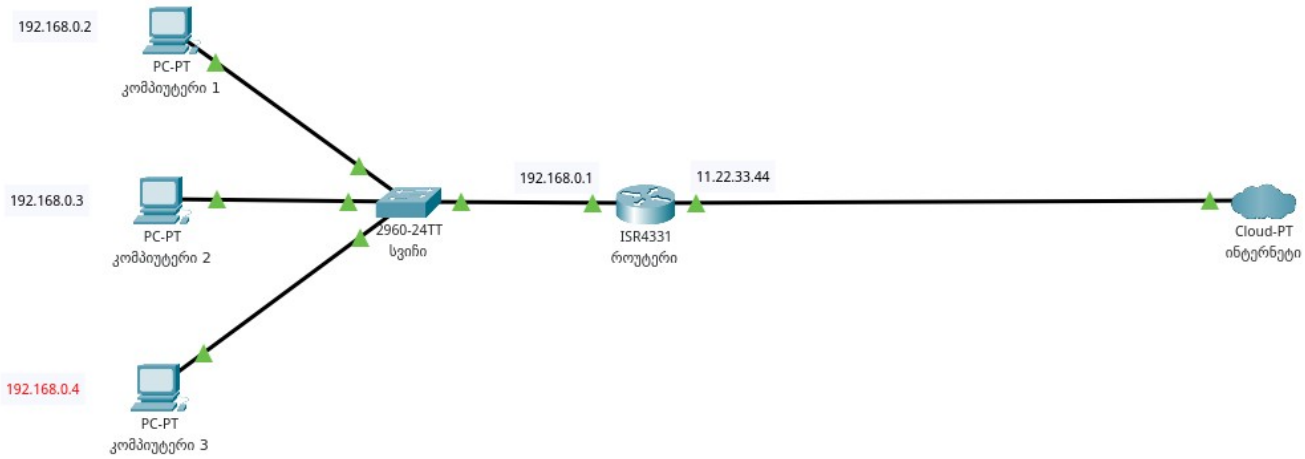
```
client, address = Server_Socket.accept()
```

აქ ორი ცვლადია მძიმით გამოყოფილი.client და address.client ცვლადში გვაქვს რამდენიმე ჰედერი კლიენტის შესახებ, ხოლო address ცვლადში კი ვიღებთ კლიენტის IP მისამართს და პორტს(საიდანაც წამოვიდა კავშირი).ეს ორი

ცვლადი გავუტოლეთ Server_Socket.accept() ფუნქციას.ეს ფუნქცია უზრუნველყოფს კლიენტების მიღებას.
შემდეგ ფუნქციაში გვიწერია clients.append(address).ეს ნიშნავს იმას, რომ clients სიაში უნდა მოხდეს შემოსული კლიენტის address ცვლადის დამატება(ანუ IP მისამართის და პორტის დამატება).ბოლოს კი ამ ფუნქციას ვიძახებთ.

თავი 12
ქსელი

ქსელის არცოდნა ძალიან ცუდია.ჩვენ მაქსიმალურად ვეცდებით ქსელის საფუძვლებში გაგარკვიოთ.ზევით შესაძლოა ჩვენ ქსელს რამდენიმეჯერ შევეხეთ, მაგრამ აქ ახლა ყველაფერს დეტალურად განვიხილავთ.გავიხსენოთ IP მისამართები.მათ იმისათვის ვიყენებთ, რომ deviceებმა გაიგონ ვის ესაუბრებიან.მაგრამ IP მისამართი არ არის ერთადერთი მისამართი.ასევე არსებობს MAC მისამართი, რომელიც ასევე ვახსენეთ ზემოთ.IP მისამართები layer 3 მისამართებია, ხოლო MAC მისამართები layer 2(განვიხილეთ მესამე თავში).ავიღოთ მარტივი ქსელის მაგალითი და განვიხილოთ:



დააკვირდით როუტერს.მას ორი IPv4 მისმართი აქვს.”11.22.33.44” არის საჯარო, რომელსაც იგი იყენებს ინტერნეტში, ხოლო “192.168.0.1” არის მისი კერძო მისამართი, რომელსაც ის იყენებს შიდა ქსელში(LAN).ალბათ გაინტერესებთ, თუ როუტერს ორი IP მისამართი აქვს, მარცხნივ მყოფ სამ კომპიუტერს რატომ აქვს მარტო ერთი?რეალურად ამ კომპიუტერებს ორი IP მისამართი აქვთ.როდესაც თქვენ როუტერს უკავშირდებით, თქვენ გეძლევათ ორი IP მისამართი: საჯარო და კერძო.საჯაროს იყენებთ ინტერნეტში, ხოლო შიდა ქსელში კერძოს.გაინტერესებთ რომელია ის საჯარო IP, რომელსაც როუტერი გაძლევთ?პასუხია ის, რომელიც როუტერს აქვს.კიდევ ერთხელ

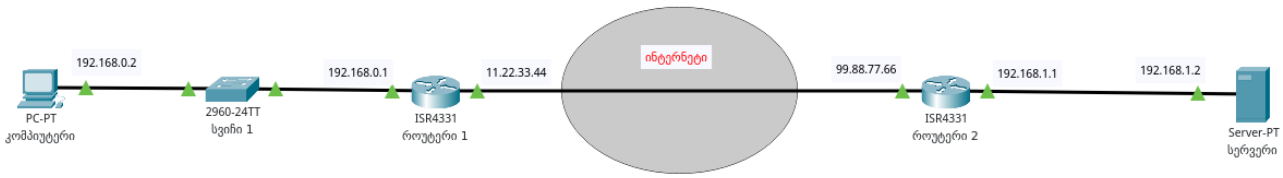
შეხედეთ სქრინშოტს.ეს სამი კომპიუტერი ინტერნეტში იყენებს ამ როუტერის IP მისამართს.ანუ სამივე კომპიუტერს ერთი და იგივე IP მისამართი აქვს ინტერნეტში.მაგრამ შიდა ქსელში მათი IP მისამართები სხვადასხვაა.ეს იმიტომ, რომ როუტერმა ისინი ერთმანეთისაგან გაარჩიოს.

ახლა ალბათ უყურებთ კომპიუტერებსა და როუტერს შორის მყოფ მოწყობილობას(სვიჩს) და გაინტერესებთ თუ რა არის ის.ეს არის layer 2 მოწყობილობა, რომელსაც აქვს ბევრი Ethernet პორტი(კაბელის შესაერთებელი) და ორი გამყვანი(Trucking) პორტი.Trucking პორტი უკავშირდება როუტერს(GigabitEthernet0/0/0 ეს არის პორტის დასახელება), ხოლო დანარჩენი პორტები უკავშირდება კომპიუტერებს.როუტერს არ აქვს იმდენი პორტი რამდენიც სვიჩს აქვს, აქედან გამომდინარე ჩვენ სვიჩის გამოყენებით ვაკავშირებთ ბევრ მოწყობილობას როუტერთან.ახლა ალბათ გაინტერესებთ თუ რატომ არ აქვს IP მისამართი სვიჩს.პასუხი ზევით უკვე ვახსენეთ.ის არის მეორე ფენაზე(layer 2) მომუშავე მოწყობილობა და მას არ აქვს მესამე ფენაზე(layer 3) წვდომა, ასე რომ მას IP მისამართი ვერ ექნება(IP მისამართი ხომ მესამე ფენის მისამართია).მაგრამ სვიჩს აქვს MAC მისამართი, რომლითაც მანიპულირებს ქსელის მეორე ფენაზე(MAC მისამართი ხომ მეორე ფენის მისამართია).ასე გამოიყურება MAC მისამართი:

00:00:5e:00:53:af

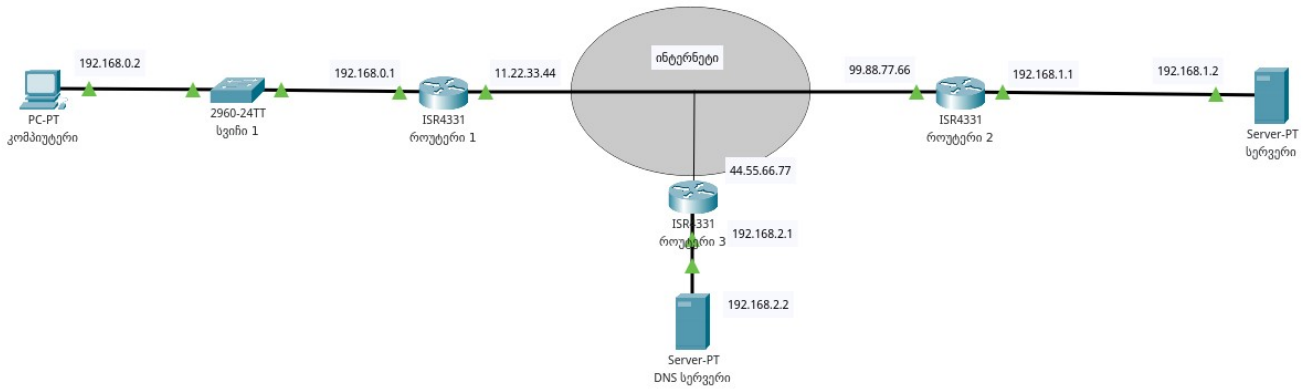
ეს არის 48 Bit MAC მისამართი(ანუ 48 ელექტრონისაგან შესდგება).ყველა ქსელურ მოწყობილობას აქვს ჩაშენებული MAC მისამართი.

მგონი ზევით მოცემული ქსელის მაგალითი გასაგებია.ახლა განვიხილოთ როგორ ხდება ინტერნეტში ორი მოწყობილობის დაკავშირება:



გამარტივებულად ასე გამოიყურება ქსელი(სერვერის მხარეს შიდა ქსელი ხშირად ძალიან კომპლექსურია).დავუშვათ ამ სერვერზე არის HTTP პორტი გახსნილი(ანუ მე-80).კომპიუტერს უნდა ამ ვებსაიტზე გადასვლა.კომპიუტერს შეუძლია ბრაუზერში ჩაწეროს: “http://99.88.77.66”, ამის შემდეგ ბრაუზერი გააგზავნის HTTP მოთხოვნას სერვერზე, სერვერი მას მისცემს html და css ფაილებს, შემდეგ მას ბრაუზერი დაარენდერებს და ეკრანზე საიტი გამოჩნდება.ადამიანისთვის ძალიან ძნელი იქნება ასე რიცხვებით იმახსოვროს მისამართები.წარმოიდგინეთ, თქვენ რომ facebook.comზე გადასვლა დაგჭირდეთ და ლინკები არ არსებობდეს, თქვენ მოგიწევდათ facebookის სერვერის IP მისამართის ჩაწერა ბრაუზერში.ამიტომ მოიგონეს დომეინები(domains).დომეინი ნიშნავს ლინკს.დედამიწაზე უამრავი საიტია და რა თქმა უნდა ყველა საიტის დომეინს ვერ შევინახავდით ყველა კომპიუტერში.ამიტომაც მოიგონეს DNS(Domain Name System) სერვერები.ეს სერვერები ძალიან გამოსადეგია და თქვენ მათთან ყოველდღე გაქვთ შეხება.როგორ მუშაობს DNS სერვერი?DNS სერვერს ჩაწერილი აქვს ლინკები და მათთან გატოლებული IP მისამართები თავის მონაცემთა ბაზაში.როდესაც

მომხმარებელი ბრაუზერში ეძებს რაიმე ვერსაიტს, თქვენი კომპიუტერი მიდის DNS სერვერთან და ეკითხებაL “უკაცრავად...www.website.comის IP მისამართი რა არის?” და DNS სერვერი გპასუხობთ: “99.88.77.66” ამის შემდეგ თქვენი ბრაუზერი ამ IP მისამართზე გადადის(კონკრეტულად პორტზე 80 ანუ HTTP ან 443 ანუ HTTPS).ქსელში ეს ასე იქნება:



მგონი ყველაფერი ნათელია.

ქვეთავი: IP მისამართები

მესამე თავში კი განვიხილეთ IP მისამართები, მაგრამ ახლა უფრო დეტალურად უნდა განვიხილოთ.დააკვირდით შიდა ქსელის IP მისამართებს.თუ როუტერის IP მისამართია “192.168.0.1” კლიენტების IP მისამართები არის: “192.168.0.2-254”.ეს იმიტომ, რომ “192.168.0” აღნიშნავს ქსელს, ხოლო ბოლო რიცხვი აღნიშნავს ჰოსტს(ქსელურ მოწყობილობას).ეს აღნიშვნა დამოკიდებულია ქსელის მასკაზე(Network Mask).

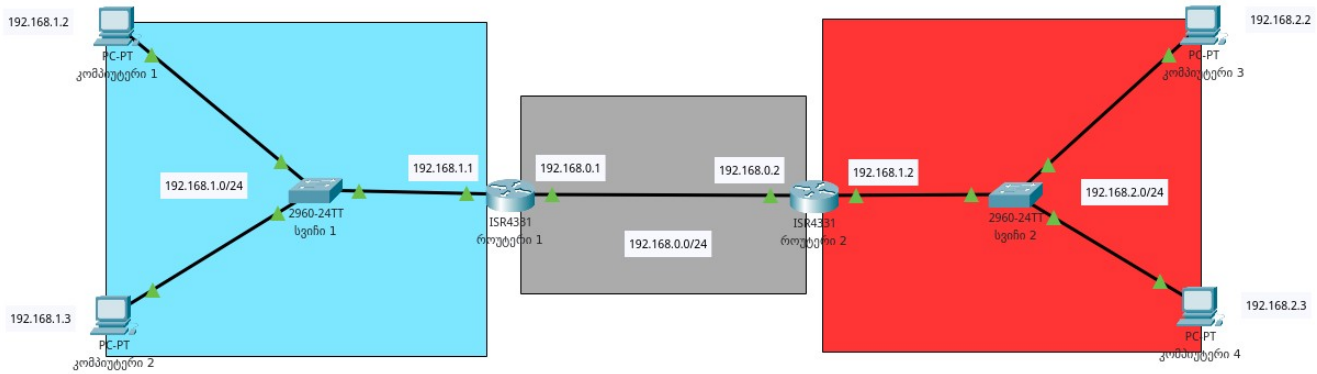
* რა არის ქსელის მასკა?
    ქსელის მასკა IP მისამართს ორ ნაწილად ჰყოფს.ზემოთ უკვე ვახსენეთ, პირველი ნაწილი ქსელს აღნიშნავს, მეორე კი ჰოსტს.

ქსელის მასკის მაგალითია:

255.255.255.0

ეს ყველაზე ხშირად გამოიყენება.ამ მასკას ეწოდება 24 ბიტიანი მასკა.ეს იმიტომ, რომ როგორც უკვე მესამე თავში ვახსენეთ IP მისამართში თითო რიცხვს აღნიშნავს 8 ბიტი.ხოლო აქ მაქსიმალურ მნიშვნელობაზე დგას სამი რიცხვი.3 * 8 = 24.ანუ ქსელის მასკა არის 24 ბიტიანი.ქსელის მასკიდან შეგვიძლია გავიგოთ IP მისამართი როგორაა დაყოფილი.ამ მასკაზე პირველი სამი რიცხვი ქსელს აღნიშნავს, ხოლო ბოლო რიცხვი ჰოსტს.აქედან გამომდინარე თუ 1 რიცხვს აღნიშნავს 8 ბიტი მაშინ რიცხვის მაქსიმალური მნიშვნელობაა 255(ესეც ვთქვით მესამე თავში).და თუ მაქსიმალური მნიშვნელობაა 255 გამოდის, რომ 24 Bit მასკიან ქსელში მაქსიმუმ 254 ქსელური მოწყობილობა შეიძლება იყოს(ანუ ამ ქსელში მაქსიმუმ 254 IP

მისამართი იქნება გაცემული).რატომ არის საჭირო ქსელების აღნიშვნა?ამაზე პასუხს გაგვცემს ეს სქრინშოტი:



როგორც ხედავთ აქ სამი სხვადასხვა ქსელი გვაქვს.პირველია 192.168.0.0/24(ანუ 24 ბიტიანი მასკით), მეორეა 192.168.1.0/24 და მესამეა 192.168.2.0/24.პირველი ქსელი(ნაცრისფერი) არის როუტერების დამაკავშირებელი ქსელი(რათა მათ შორის ინფორმაცია გაიცვალოს).მეორე არის ცისფერი ქსელი 192.168.1.0/24 სადაც შეგვიძლია გვქონდეს 255 მოწყობილობა(როუტერის გარდა 254), ხოლო მესამეა წითელი 192.168.2.0/24 და აქაც შეგვიძლია 255 მოწყობილობა გვქონდეს.მოწყობილობის ლიმიტი იმიტომ გვაქვს, რომ გვაქვს IP მისამართების ლიმიტი(უკვე ავხსენით მესამე თავში თუ რატომაა ასე).ავიღოთ მაგალითად “კომპიუტერი 2”ის IP მისამართი და განვიხილოთ:

192.168.1.3

პირველი სამი რიცხვით იმას ვიგებთ თუ რომელ ქსელშია მოწყობილობა.ბოლო რიცხვით კი ვიგებთ ქსელში მის კონკრეტულ მდებარეობას.ისევ სამეზობლოს მაგალითი ავიღოთ.მაგალითად თქვენ ცხოვრობთ იოსებ სტალინის N3 სახლში.წარმოიდგინეთ, რომ “192.168.1” ნიშნავს “იოსებ სტალინის”ს, ხოლო “3” ნიშნავს “N3”ს.ახლა იმედია ყველაფერი გასაგებია.

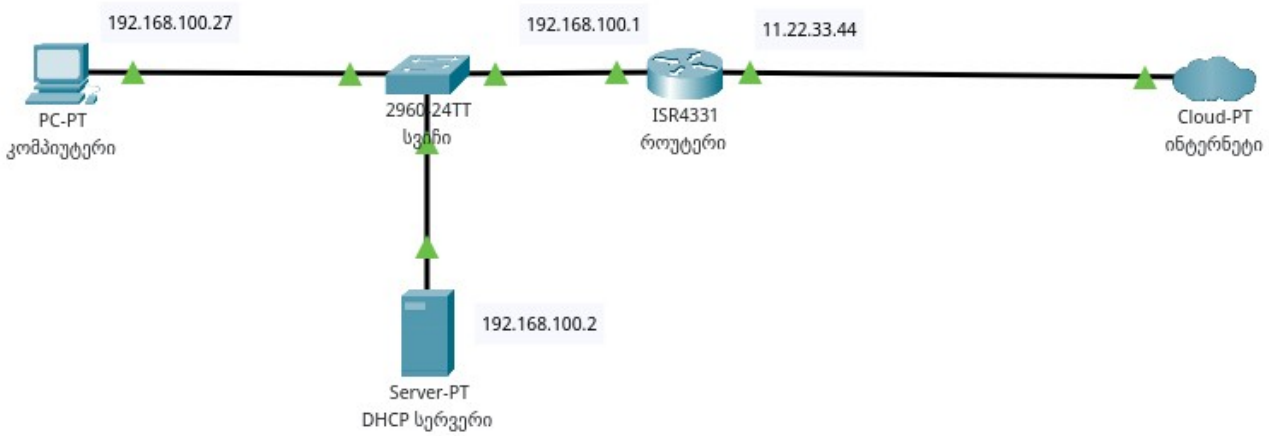
ქვეთავი: DHCP

DHCP(Dynamic Host Configuration Protocol) არის პროტოკოლი, რომელიც ქსელში ავტომატურად ანაწილებს IP მისამართებს.ზოგადად IP მისამართი არის სტატიკური და დინამიური.სტატიკური IP მისამართი არის ისეთი მისამართი, რომელიც არ იცვლება, ხოლო დინამიური IP მისამართი არის ისეთი მისამართი, რომელიც რაღაც დროის შემდეგ შეიცვლება.არსებობენ DHCP სერვერები, რომლებიც როდესაც დაინახავენ, რომ ქსელში ახალი მოწყობილობა ჩაერთო, ის ავტომატურად მიაბამს ამ მოწყობილობას IP მისამართს.ეს ბევრ რამეში გვეხმარება.მაგალითად თქვენ ქსელის ინჟინერი ხართ რომელიმე დიდ კომპანიაში.მათ შენობებში ძალიან დიდი და კომპლექსური ქსელებია.თქვენ დაიტანჯებოდით ყველა მოწყობილობისათვის

რომ ცალ-ცალკე მიგენიჭებინათ IP მისამართები.ამიტომ გამოიყენება DHCP სერვერები.

* რა ხდება სახლის ქსელის შემთხვევაში?
    სახლის როუტერებს DHCP სერვისი ჩაშენებული აქვთ, ასე რომ როდესაც თქვენი სახლის როუტერს დაუკავშირდებით, ის IP მისამართს ავტომატურად მოგანიჭებთ.მაგალითად თუ თქვენი სახლის IP მისამართი LANში არის “192.168.100.1” თქვენ ის მოგცემთ მისამართს “192.168.100.5”.

ესაა ქსელში DHCP სერვერის მაგალითი:



DHCP სერვერმა მისცა კომპიუტერს შიდა ქსელში IP მისამართი “192.168.100.27”.ახლა კომპიუტერს თავისუფლად შეუძლია ინტერნეტში ნებისმიერი რამ გააკეთოს.

აქ დავამთავროთ ქსელზე საუბარი, მაგრამ მომავალ თავებში მას კიდევ შევეხებით.

თავი 13
Introduction to Cyber Attacks

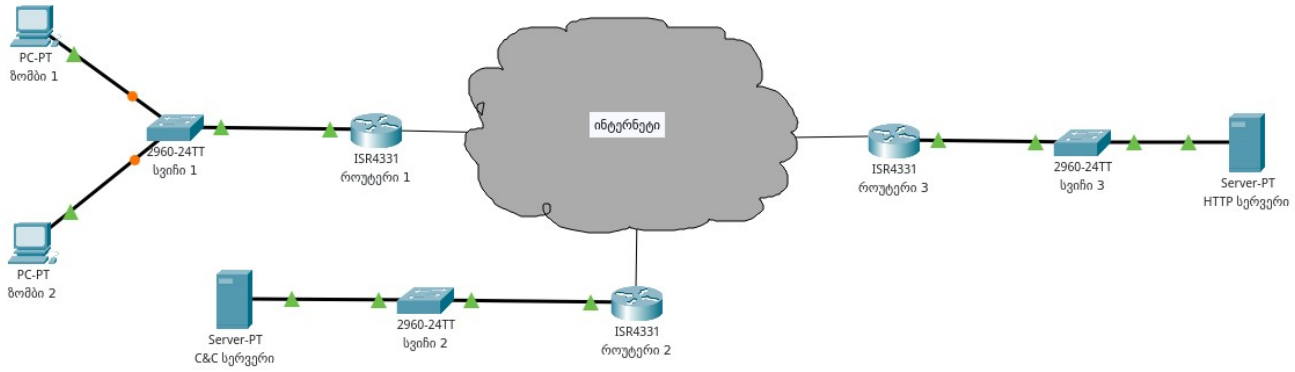
კიბერთავდასხმები საკმაოდ მომრავლებულია მსოფლიოში.სტატისტიკის მიხედვით კიბერომი ჩვეულებრივ ომზე სახიფათოა(ბიუჯეტის დაზიანების მხრივ).ჩვენ ახლა განვიხილავთ რამდენიმე კიბერშეტევის სახეობას და მათზე ვიმსჯელებთ.

ქვეთავი: DOS(Denial of service)

DOS შეტევა ზევითაც განვიხილეთ(როდესაც საუბარი იყო პაკეტებზე).მაგრამ ახლა უფრო დეტალურად განვიხილოთ.DOS შეტევა არის რაიმე სერვისის დაბლოკვის საშუალება.მაგალითად სერვერზე თუ არის HTTP პორტი გახსნილი და ჩვენ გვინდა, რომ ეს ამ პორტზე კლიენტი ვერ შევიდეს(ანუ ვებსაიტზე ვერავინ გადავიდეს) ჩვენ ვიწყებთ flood შეტევას.ეს იმას ნიშნავს, რომ ჩვენ ძალიან ბევრ პაკეტს გავაგზავნით ამ პორტზე და სერვერს არ ექნება იმის საშუალება, რომ სხვა კლიენტებს მოემსახუროს.ის დაკავებული იქნება ჩვენით.სხვა კლიენტებს ბრაუზერი დაუწერს, რომ ვებსაიტი მიუწვდომელია.არის რაღაც ხელსაწყოები, რომლებიც DOSის საშუალებას გვაძლევს, მაგრამ თუ ოდნავი კოდირება მაინც იცის ადამიანმა, ის შეძლებს და ააწყობს საკუთარ DOS ხელსაწყოს.

ქვეთავი: DDOS(Distributed denial of service)

DDOS შეტევა არის იგივე რაც DOS შეტევა, მაგრამ აქ რამდენიმე განსხვავებაა.DDOS შეტევას ასრულებს ე.წ “ბოტნეტი”(Botnet).ბოტნეტი არის “ზომბი” კომპიუტერების არმია, რომლებიც ემორჩილებიან C&C(Command and control) სერვერს.ამ სერვერს კი ბრძანებას ჰაკერი აძლევს.ზომბის შექმა ძალიან ადვილია.კომპიუტერში უნდა შეაპაროთ კოდი, რომელიც მას ზომბად აქცევს.ეს კოდი რომ დეტალურად ავღწეროთ ასეთი იქნება: კოდი კომპიუტერზე გააღებს პორტს და დაელოდება ბრძანებას, როდესაც C&C სერვერი ამ პორტს ინტერნეტში დააბროდკასტებს(broadcast), ყველა ზომბს მიუვა სამიზნის ინფო და ისინი დაიწყებენ მასზე DOS შეტევას.შესაძლოა ბოტნეტში იყოს 12 000 ზომბიც კი(ეს კიდევ ცოტაა).წარმოიდგინეთ რა მოხდება, როდესაც 12 000 ზომბი წამში 1 000ჯერ მაინც გააგზავნის სერვერის რომელიმე სერვისზე, მაგალითად FTPზე მოთხოვნას.ამ სერვისზე ჩვეულებრივი კლიენტი ვეღარ შევა.ბოტნეტი ქსელში დაახლოებით ასე გამოიყურება:



C&C სერვერი გააკონტროლებს ზომბებს და უბრძანებს, რომ შეუტიონ HTTP სერვერს.

ქვეთავი: MITM(Man in the middle)

MITM შეტევა გამოიყენება ინფორმაციის მოსაპარად.მაგალითად LANში გვაქვს სახლის როუტერი, რომლის IP მისამართიც არის “192.168.100.1”.არის

კომპიუტერი ამ სახლში, რომლის IP მისამართიც არის “192.168.100.2”.ამ კომპიუტერიდან შედიან ონლაინ კაზინოში.ჩვენ გვინდა, რომ ამ ონლაინ კაზინოს login და password გვქონდეს(ალბათ იმიტომ, რომ ბლეკჯეკში სხვისი ფული წავაგოთ და არა ჩვენი).ჩვენ ჯერ უნდა დავუკავშირდეთ ამ როუტერს(ალბათ Wifiს გატეხვა მოგიწევთ, რომელსაც ვახსენებთ ამ წიგნში).დაკავშირების შემდეგ ჩვენი IP მისამართი დავუშვათ არის “192.168.100.3”.ამის შემდეგ ჩვენ ვიწყებთ MITM შეტევას.MITM შეტევა მოქმედებს შემდეგნაირად.ზოგადად კომპიუტერები კითხულობენ თუ რომელი არის როუტერი.შემდეგ როუტერი ეუბნება მათ, რომ ის არის როუტერი.ჩვენ შეგვიძლია ჩვენი კომპიუტერიდან ისეთი პაკეტები გავაგზავნოთ, რომ სამიზნე კომპიუტერი დაიჯერებს იმას, რომ ჩვენ ვართ როუტერი.ანუ პაკეტებს ჯერ ჩვენ გამოგვიგზავნის და ჩვენ მათ ნამდვილ როუტერზე გადავამისამართებთ.ანუ ჩვენ ახლა ტრაფიკს “ვკითხულობთ”.ჩვენ შეგვიძლია Wiresharkით ჩავიწეროთ ქსელის ტრაფიკი .pcap ფაილში და შემდეგ მისი ანალიზი გავაკეთოთ.როდესაც სამიზნე კომპიუტერი ონლაინ კაზინოს საიტზე გააგზავნის loginსა და passwordს ეს პაკეტები ჯერ ჩვენს კომპიუტერში გაივლის და შემდეგ ჩვენ როუტერს გადავაწვდით.ანუ ჩვენ უკვე ვიცით სამიზნის ონლაინ კაზინოს ექაუნთი(account).ამის შემდეგ უბრალოდ შევდივართ მის ანგარიშზე და ვიწყებთ ბლეკჯეკში ფულის წაგებას(კაზინოს ვერ მოუგებთ).

ქვეთავი: Phishing

ბავშვობაში თუ გამოგიყენებიათ საიტი zshadow?თუ კი, გილოცავთ!თქვენ კიბერთაღლითობაში გაქვთ მიღებული მონაწილეობა(რაც კანონით trust us...დასჯადია).ფიშინგი არ არის მხოლოდ ლინკის დაგენერირება და გაგზავნა.შესაძლოა ეს იყოს კარგად შენიღბული mail, რომელზეც წამოეგება სამიზნე და მოგვცემს საჭირო ინფორმაციას.ფიშინგი(phishing) დღესაც საკმაოდ გავრცელებულია.როდის იყენებენ ფიშინგს?მაგალითად თქვენი სამიზნეა რომელიმე კომპანიაზე წვდომის აღება.იცით, რომ მათ ოფისში ზის მოხუცი ქალბატონი(კომპიუტერთან მუშაობს), რომელიც არის ზედმეტად ბრიყვი(არ გვინდოდა ამ სიტყვის გამოყენება.გვაპატიეთ).ასევე მას უყვარს კატები.თქვენ მეილზე უგზავნით შეტყობინებას, რომ თუ გადავა რაღაც კონკრეტულ ლინკზე ის კატების ფოტოებს დაათვალიერებს.ქალბატონიც სიხარულით გადადის ამ ლინკზე.ეს ლინკი თქვენი შექმნილია და ქალბატონი გადაყავს საიტზე, რომელიც კლიენტს ატყუებს, რომ მისი კომპიუტერი დავირუსებულია და მან SSH login და password უნდა ჩაწეროს, რათა კომპიუტერში IT თანამშრომელი შევიდეს და ვირუსი წაშალოს.ამ ქალმა არ იცის რა არის SSH მაგრამ კომპიუტერის ეკრანზე წერია მისი login და password.ქალი შეშინებულია და არ უნდა რომ კომპიუტერის “დაზიანებისთვის”(ბრჭყალებში იმიტომ ვწერთ, რომ მას რეალურად არაფერი გაუფუჭებია...ჯერ) დასაჯონ, ასე რომ მას ეს მონაცემები შეჰყავს.საიტი ამ მონაცემებს თქვენ გიგზავნით.შემდეგ უბრალოდ თქვენ უნდა წახვიდეთ კომპანიის შიდა ქსელში შეხვიდეთ, იპოვოთ იმ ქალბატონის კომპიუტერის IP მისამართი და SSHს დაუკავშირდეთ.ამის შემდეგ იმ კომპიუტერიდან ამოიღებთ სენსიტიურ ინფორმაციას და ბედნიერი წახვალთ სახლში, რათა სხვისი ონალინ კაზინოს ექაუნთით ბლეკჯეკში ფული წააკოთ(წინა ქვეთავში ვახსენეთ).

ქვეთავი: Ransomeware(N3tR3tის საყვარელი ქვეთავი)

რა არის Ransomeware?ოდესმე გაგიგიათ ვირუსზე “WannaCry” ან “NotPetya”?ეს ორი ვირუსი არის რენსამვეარი(Ransomeware).რენსამები ძალიან ბოროტი დემონები არიან.ისინი გენიოსებმა შექმნეს, რათა ფული ეშოვათ.კონკრეტულად WannaCry ვირუსის შემქმნელმა ამ შეტევით მსოფლიოს 4 მილიარდი ამერიკული დოლარი მოპარა.

რენსამვეარის მუშაობის პრინციპი მარტივია.როდესაც ის კომპიუტერში მოხვდება, ის აენკრიპტებს(Encrypt) ანუ შიფრავს ფაილებს.შემდეგ ის კომპიუტერს აიძულებს, რომ ყოველი ჩართვისას ვირუსი გაუშვას.ანუ ამ ვირუსისგან ვერ გათავისუფლდებით სანამ იმას არ გააკეთებთ რასაც ის გეტყვით.თუ გინდათ, რომ თქვენი ფაილები გადარჩეს, რენსამები გიწერენ ბიტკოინის მისამართს და ამბობენ, რომ როდესაც გადარიცხავთ გარკვეულ თანხას(WannaCry 600$ს ითხოვდა) მაშინ მიიღებთ “გასაღებს”, რომელიც ფაილების დადეკრიპტებაში(Decrypt) ანუ გაშიფრვაში დაგეხმარებათ და ვირუსიც მოგშორდებათ.ყველაზე ცუდი ისაა, რომ რენსამებს აქვთ დროის ლიმიტი.თუ გარკვეულ დროში თანხას არ გადარიცხავთ, მაშინ რენსამი თქვენს დისკს დააფორმატებს(ანუ ყველაფერს წაშლის სამუდამოდ).ასეთი ვირუსები დაწერილია C პროგრამირების ენაში და არის იდეალურად გათვლილი.C პროგრამირების ენაში იმიტომ, რომ უკვე გადაკომპლირებული კოდის დარევერსება(reverse) ანუ შეტრიალება ურთულესია და თუ ადამიანი პროფესიონალი არაა ის ვერ გაიგებს გადაკომპილირებული კოდი რას აკეთებს(ამას reverse ინჟინერია ქვია).

ქვეთავი: ჯაშუშები(D1sM3 ამათი ფანია)

ჯაშუშები ძალიან მაგარი ვირუსებია.ისინი ყველაფერს იწერენ რასაც მომხმარებელი კომპიტერში აკეთებს და რაღაც დროის მონაკვეთში აგზავნიან ჰაკერთან.განვიხილოთ ჯაშუშები:

1) Keylogger – ქილოგერი(Keylogger) იწერს ყველაფერს რასაც კი აკრიფავთ კლავიატურაზე და შემდეგ აგზავნის ჰაკერთან.
2) WebCamSpy – ვებკამერის ჯაშუში იწერს თქვენი კომპიუტერის კამერიდან გამოსახულებას და აგზავნის ვიდეოს ფორმატით ჰაკერთან ან თუ ვიდეოს ფორმატით არ იწერს მაშინ პირდაპირი ნაკადით აწვდის ჰაკერს(live stream).
3) MicroSpy – მიკროფონის ჯაშუში იწერს კომპიუტერის მიკროფონიდან ხმას და აგზავნის ჰაკერთან აუდიო ფორმატში.

ჯაშუშები ძალიან რთული შესამჩნევია, რადგან ისინი ტროიანი(Trojan) ვირუსებია.ტროიანი, იგივე ტროას ცხენი, არის ვირუსი, რომელიც ტროას ცხენის პრინციპით მუშაობს.მაგალითად კომპიუტერს ატყუებს, რომ ის არის .png ფაილი(ფოტო) და როდესაც კომპიუტერი მას ჩაიწერს ის მავნე საქმიანობას ეწევა.

ქვეთავი: CryptoJacking

იცით როგორ გამოიმუშავებენ კრიპტოვალუტას?კომპიუტერები მათ “ამაინინგებენ”(mining).მაინინგი არის საკმაოდ რთული პროცესი.ამ დროს კომპიუტერი სრულ დატვირთვაზეა და ასრულებს ძალიან ბევრ და რთულ მათემატიკურ კალკულაციას ძალიან სწრაფად.არსებობს მაინინგ ფერმები სადაც დგას უმძლავრესი დანადგარები, რათა ბიტკოინი ან სხვა კრიპტოვალუტა გამოიმუშავონ, მაგრამ ეს დანადგარები ძალიან ძვირია.

როგორც ხედავთ მაინერი საკმაოდ ძვირი ღირს.ამ კონკრეტულის ფასი გახლავთ 9 585 ამერიკული დოლარი(ამაზონზე).ასეთი კი მაინინგ ფერმებში უამრავი დგას.ისინი ძალიან დიდ ელექტროენერგიას მოიხმარენ და ყოფილა შემთხვევები, როდესაც ასეთ ფერმაში დენის გადასახადი 50 000 ამერიკული დოლარიც მისვლიათ.ბევრ ჰაკერს ასეთი დიდი ფული არ გააჩნია.აქედან გამომდინარე ისინი იწყებენ მავნე პროგრამების შექმნას სახელად CryptoJacker.ასეთი ვირუსები ტროიანის პრინციპით ძვრებიან კომპიუტერებში და ჩუმად იწყებენ კრიპტოვალუტის გამომუშავებას თქვენი კომპიუტერის ხარჯზე.როდესაც ჩართავთ კომპიუტერს ის ავტომატურად დაიწყებს მაინინგს და თქვენს კომპიუტერს საკმაოდ მალე პრობლემები შეექმნება.თანაც როდესაც კომპიუტერი სრულ დატვირთვაზე მუშაობს და დამატებით თქვენ მისცემთ რაიმე დავალებას ის გაჭედავს და თქვენ ეს ძალიან არ მოგეწონებათ.ალბათ შემდეგ თუ თქვენს კომპიუტერს სათანადოდ არ მიხედეთ გაფუჭდება.სამაგიეროდ ეს ვირუსი თქვენი კომპიუტერით ფულს გამოიმუშავებს და ჰაკერს უგზავნის.ანუ თქვენ ორმაგად ზიანდებით.პირველი ზიანი ისაა, რომ კომპიუტერი გაგიჭედავთ, მეორე ზიანი ისაა, რომ თქვენს ხარჯზე ფულს გამოიმუშავებენ და დიდი ალბათობით ასეთი დატვირთვა თქვენს კომპიუტერს დააზიანებს.

თავი 14
ანონიმურობა ინტერნეტში

ინტერნეტში ანონიმურობა ყველას უნდა.მაგრამ ხშირად ეს არასწორად ესმით.არის რაღაც გზები, რომლებითაც შესაძლოა ანონიმურობა მოიპოვოთ ინტერნეტში, მაგრამ რეალური გზა გაცილებით რთულია.არსებობენ VPNები(Virtual Private Network).ალბათ თქვენც გამოგიყენებიათ ეს IP მისამართის შესაცვლელად.VPNების პრობლემა ისაა, რომ მათ logging სისტემა აქვთ და ყველაფერს იწერენ.ანუ იწერენ მათ ვინ დაუკავშირდათ და რა გააკეთეს კავშირის შემდეგ.ანუ თქვენ, რომ VPNით პენტაგონი გატეხოთ, პენტაგონი მიაკითხავს VPN პროვაიდერს და ეტყვის logging ფაილები გვაჩვენეთო.შემდეგ იპოვიან თქვენს IP მისამართს და მივლენ თქვენს პროვაიდერთან.თქვენი პროვაიდერი აჩვენებს თუ ვის ეკუთვნოდა იმ მომენტში IP მისამართი.აღმოჩნდებით თქვენ და ასე ადვილად დაგიჭერენ.

მეორე გზა არის Proxy სერვერები.კარგი ისაა, რომ თქვენ შეგიძლიათ Proxy სერვერების ჯაჭვი(chain) გააკეთოთ და რამდენიმეჯერ გადაამისამართოთ IP მისამართი:

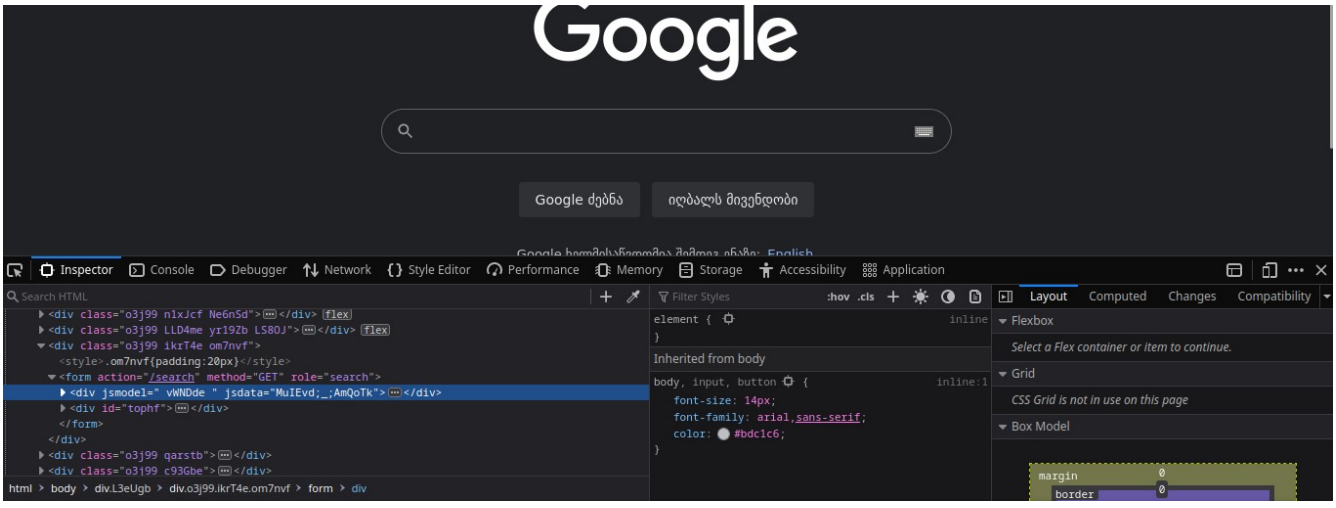
მაგალითად ნახატზე მოცემული შავი წერტილი ხართ თქვენ, წითელი თქვენი სამიზნეა, ხოლო ლურჯი წერტილები Proxy სერვერებია.ამ შემთხვევაში თქვენი IP მისამართი და ლოკაცია 4ჯერ შეიცვლება.როდესაც პენტაგონი თქვენს IP მისამართს გამოყვება ისინი ჯერ პანამის არხთან მყოფ პროქსი(Proxy) სერვერს იპოვიან, შემდეგ ისევ გამოყვებიან ხაზს და ბრაზილიაში იპოვიან სერვერს, შემდეგ აფრიკაში, მავრიტანიაში აღმოაჩენენ სერვერს, შემდეგ ჩინეთში და

ბოლოს საქართველოში ჩამოვლენ.ამ დროისათვის თქვენ ქვეყნიდან გაქცევა და დამალვა უნდა შეძლოთ.საუკეთესო გზა ამერიკისგან თავის გადასარჩენად რუსეთია(მაგრამ ახლა უკრაინაში ომია და რუსეთში ვერ შეხვალთ).

საუკეთესო გზა დასამალად მაინც Tor Networkში შესვლა გვგონია.ეს არის იდეალურად მოფიქრებული ქსელი.აქაა ე.წ. დარკნეტი/დარკვები(DarkNet/DarkWeb).აქ ანონიმურობა სრულიად დაცულია, მაგრამ 100%იან გარანტიას მაინც არავინ მოგცემთ.სამწუხაროდ, თქვენ თუ არ გაქვთ სწრაფი ინტერნეტი torში ძრომიალი გაგიჭირდებათ.საქმე ისაა, რომ დღეს პროვაიდერებიც კი არ ინდობიან.ინტერნეტი ძალიან არასანდოა.

თავი 15
Introduction to Web

ვები(Web) ერთ-ერთი ყველაზე აუცილებელი თემაა ITში.ამ თავში თქვენ ისწავლით თუ რა არის HTTP/HTTPS, რა განსხვავებაა მათ შორის, გაიგებთ HTTP რექუესთებზე(request) და ცოტაოდენ ვებ-ჰაკინგ ხრიკებსაც ისწავლით.ვებს აქვს ორი მხარე: Front-End(კლიენტის მხარე) და Back-End(სერვერის მხარე).Front-End არის ის, რასაც კლიენტი სერვერზე გადასვლისას ხედავს ანუ დარენდედრებულ HTML და CSS კოდს.თუ თქვენ შეხვალთ ნებისმიერ საიტზე და ჩართავთ “Inspect element”ს თქვენ დაინახავთ “უცნაურ” კოდებს:



ეს არის HTML.HTML არის ენა(არა პროგრამირების), რომლითაც იწყობა ვებსაიტების წინა მხარე.წინა მხარეში შედის “Google ძებნა” ღილაკი, ტექსტის ჩასაწერი ბარი(bar), Google ლოგო და ა.შ.

Back-End არის სასერვერო მხარე(ჯოჯოხეთი დედამიწაზე).ეს ნიშნავს სერვერების მონაცემთა ბაზებზე მიბმას, სერვერების და მონაცემთა ბაზების კონფიგურაციას, სერვერის გამართვა ისე რომ კლიენტს სწორად მოემსახუროს, უსაფრთხოების სისტემის აწყობა და დაყენება და ასე შემდეგ.

მოდით ახლა ვისაუბროთ თუ როგორ მუშაობს HTTP სერვისი.მაგალითად ავიღოთ apache2 ვებ სერვერი.როდესაც apache2 სერვისს ჩავრთავთ, ის ხსნის

მე-80(HTTP) პორტს და კონკრეტული ფოლდერიდან(/var/www/html/) კითხულობს html და css ფაილებს(index.html, index.css და ა.შ.).ამ ფაილებში წერია ის კოდები რაც ზევით იყო ნახსენები.როდესაც კლიენტი შევა apache2ის მიერ გახსნილ პორტზე სერვერი მას მისცემს ამ ფაილებს, რათა კლიენტმა ამოიკითხოს კოდი ფაილებიდან და ბრაუზერში დაარენდეროს(გამოსახოს).მაგრამ ასე უბრალოდ არ გასცემს სერვერი ფაილებს.კლიენტმა უნდა აუხსნას სერვერს თუ რომელი ფაილი უნდა.ამისათვის არსებობენ HTTP მეთოდები.ახლა ჩამოვთვლით რამდენიმე მათგანს: GET, POST, PUT, DELETE და ასე შემდეგ.

1) GET – ამ მეთოდით ვიღებთ სერვერიდან საჭირო html ფაილებს.
2) POST – ამ მეთოდით ვაგზავნით სერვერზე ინფოს(login და password).
3) PUT – ამ მეთოდით შეგვიძლია რაიმე ფაილი ავტვირთოთ სერვერზე.
4) DELETE – ამ მეთოდით შეგვიძლია არსებული ფაილი წავშალოთ სერვერიდან.

ანუ თუ მომხარებელი შედის youtube.comზე ის ამ სერვერს უგზავნის HTTP GET მოთხოვნას და მისგან იღებს ყველა საჭირო ფაილს, რათა დაარენდეროს ბრაუზერში(მათ შორის ფოტოები ანუ საიტის ლოგო და ა.შ.).

მაგალითად თქვენ გინდათ facebook.comზე loginის გავლა.თქვენ შეგყავთ თქვენი მეილი და პაროლი და აწვებით “შესვლა”ს.ამ ღილაკს რომ დააწვებით, თქვენი კომპიუტერიდან იგზავნება HTTP POST მოთხოვნა, სადაც წერია თქვენი მეილი და პაროლი.სერვერი მიიღებს ამ მეილს და პაროლს, გადაამოწმებს მონაცემთა ბაზაში და თუ ვალიდურია მაშინ თქვენს ექაუნთზე შეგიშვებთ.

ალბათ გაინტერესებთ არ შეიძლება ვების წინა მხარეს პროგრამირების ენა ჰქონდეს ჩაშენებული?პასუხია კი!ჩვენ გვაქვს ასეთი ენა!მას ჰქვია ჯავასკრიპტი(Javascript).ჯავასკრიპტი არის High level ინტერპრეტირებული ენა.ისინი საიტებს უკეთ ფუნქციონირებაში ეხმარება.მაგალითად საიტს თუ უნდა, რომ დალოგოს(log) რაიმე კონსოლში ის გამოიძახებს ჯავასკრიპტს და ჯავასკრიპტში ჩაწერს ამას:

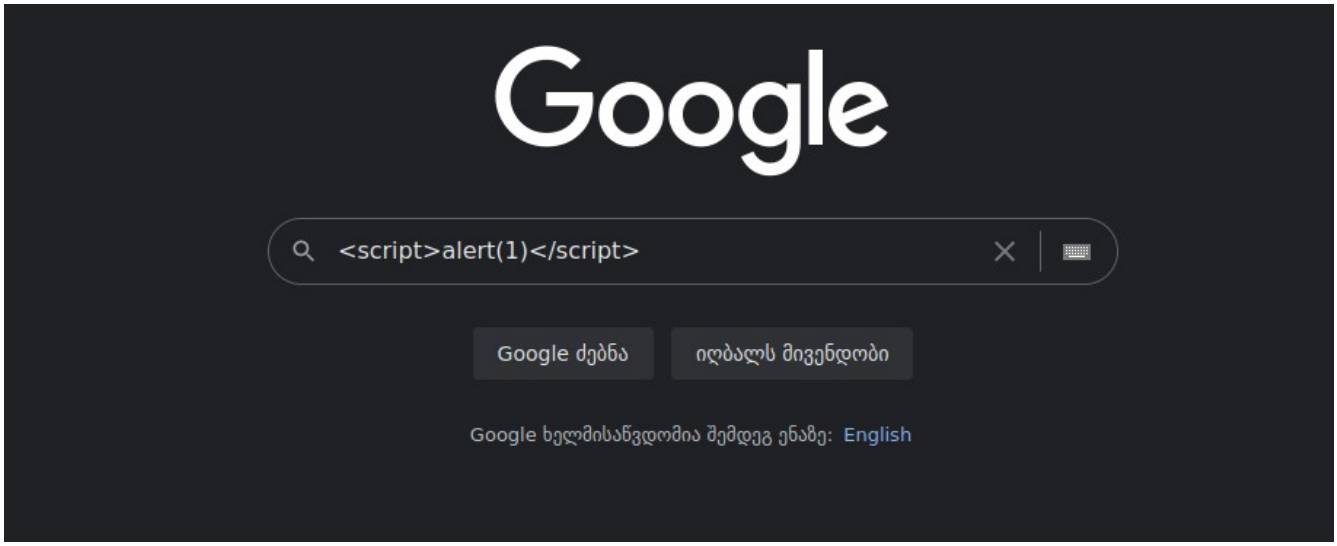
```
console.log(“აქ რამე ტექსტი”)
```

ჯავასკრიპტში log() ფუნქცია ემსგავსება პითონის print() ფუნქციას.თუ მაგალითად გვინდა, რომ საიტზე გადასვლისას ეკრანზე ამოვარდეს popup, მაშინ საიტში ჩავაშენებთ კოდს:

```
alert(“აქ რამე ტექსტი”)
```

ეს ტექსტი მომხმარებლის ეკრანზე გამოხტება.

ვებშიც არის მოწყვლადობები, რომლებსაც ჰაკერები აექსპლოიტებენ.მაგალითად ავიღოთ XSS(Cross site scripting) შეტევა.ამ შეტევის დროს ჰაკერს input გრაფაში შეჰყავს ისეთი კოდი, რაც გამოიწვევს სერვერის დაზიანებას.მაგალითად:



ეს კოდი, რა თქმა უნდა, googleზე არ იმუშავებს, მაგრამ არის ვებსაიტები, რომლებზეც ამართლებს.ახლა მოდით განვიხილოთ ეს დაწერილი კოდი.

1) <script> - ეს html ენაში ნიშნავს ჯავასკრიპტ კოდის დაწყებას.
2) alert(1) – ეს უკვე ჯავასკრიპტ კოდია რომელიც ეკრანზე გამოიტანს popupს, სადაც ეწერება რიცხვი 1.
3) </script> - ეს html ენაში ნიშნავს ჯავასკრიპტ კოდის დახურვას.

ანუ htmlში იწერება ტეგები.ტეგები არის მაგალითად: <html>, <script>, <style>, <body>, <head> და ასე შემდეგ.ყველა ეს ტეგი იხურება.შემდეგ გახსნასა და დახურვას შორის უკვე იწერება ის, რაც გვინდა რომ მათ შორის ჩავწეროთ.ჩვენ, რომ ზევით ნაჩვენებ სქრინშოტში გაჩვენეთ კოდი, ეგეც ნიშნავს ჯავასკრიპტ ტეგის გახსნას, შემდეგ ჯავასკრიპტ კოდი დავწერეთ და ბოლოს ჯავასკრიპტ ტეგი დავხურეთ.როდესაც დავაწვებით “ძებნა”ს სერვერი ამას აღიქვამს როგორც თავის კოდს და იმას გააკეთებს რაც ჯავასკრიპტ ტაგებს შორის ეწერა(ეს googleზე არ ამართლებს!).

მეორე სახის შეტევა(იგივე პრინციპი) არის SQL ინჯექტირება(Injection).თუ მონაცემთა ბაზა არასწორადაა დაკონფიგურირებული, ჩვენ შეგვიძლია მას რაღაც ისეთი გადავაწოდოთ, რომ მან სენსიტიური ინფო მოგვცეს.სანამ ამას ავხსნით, მანამდე უნდა გაიგოთ თუ როგორ მუშაობს მონაცემთა ბაზა.მონაცემთა ბაზა პროგრამირდება SQL პროგრამირების ენაში.ის ძალიან ადვილია.მონაცემთა ბაზებში გვაქვს მაგიდები(tables) სადაც არის დამატებული სხვადასხვა ცვლადები და მათი მნიშვნელობები.თუ ჩვენ ჩავწერთ საძიებო გრაფაში ასეთ კოდს:

“ AND SELECT password FROM passwdTable

და დავაწვებით “ძებნა”ს, სერვერი ამას მონაცემთა ბაზაში წაიღებს გადასამოწმებლად.როდესაც ეს კოდი მონაცემთა ბაზაში შევა და საძიებო კოდში ჩაჯდება, მოხდება შეცდომა, რადგან თუ ეს ტექსტი უნდა ჩაწერილიყო ბრჭყალებში, მაგრამ მას წინ ერთი ბრჭყალი უწერია, მონაცემთა ბაზა იფიქრებს რომ ბრჭყალები ამ ტექსტის წინ მდებარე ბრჭყალთან დაიხურა და შემდეგ

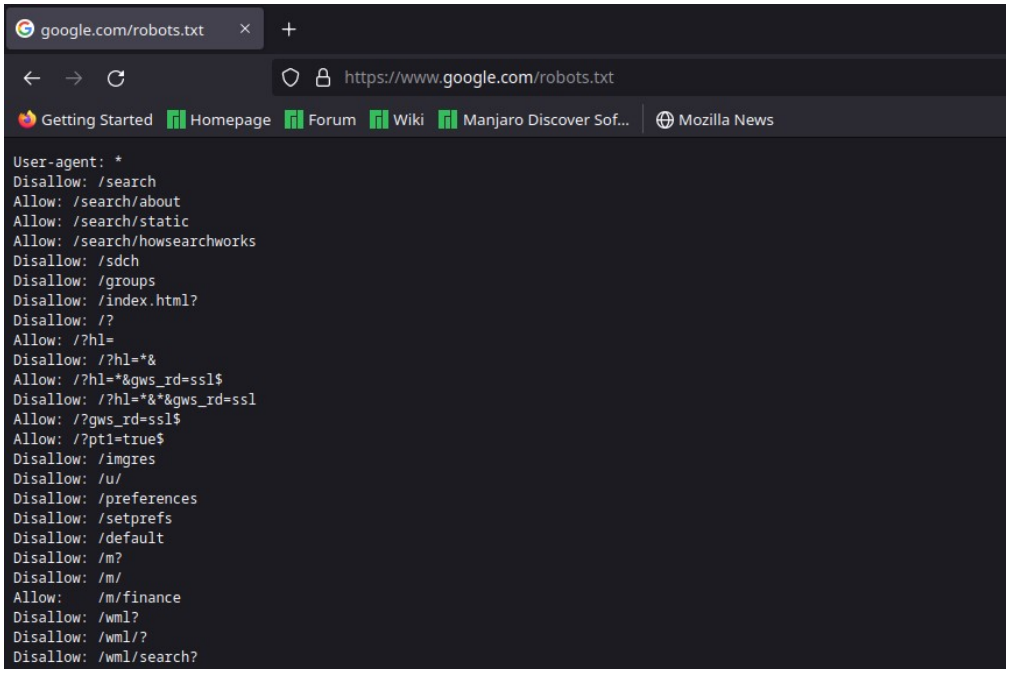
დაწერილ ტექსტს, როგორც SQL ენად ისე აღიქვამს.ჩვენ შეგვიძლია მივუთითოთ ნებისმიერი რამ და ის ამაზე პასუხს გაგვცემთ.მაგალითად ზემოთ დავწერეთ, რომ აარჩიოს პაროლები პაროლთა მაგიდიდან.როდესაც ეს მოხდება, ის პაროლებს ბრაუზერში გამოგვიტანს.

ექსპლოიტებზე საუბარი აღარ გვინდა.მოდით ვისაუბროთ ყველასათვის საყვარელ ქუქიებზე(Cookie).რა არის ბრაუზერ ქუქი(Browser Cookie)?ქუქი არის ჯავასკრიპტ კოდი, რომელსაც სერვერები გიგზავნიან.თუ თქვენ მათ მიღებას დათანხმდებით ისინი თქვენს კომპიუტერში შეინახება, ხოლო თუ უარს იტყვით მაშინ სერვერები არ მოგაწვდიან ქუქიებს(ის კონკრეტული სერვერები, რომლებსაც უარი უთხარით).ქუქიები ვებსაიტებს გამართულად მუშაობაში ეხმარება, მაგრამ არის მომენტები, როდესაც ქუქიებს ცუდად იყენებენ.აი მაგალითად შემთხვევით კომპიუტერში +18 ფოტოები ხომ არ ვარდებაა(ან ტელეფონში)?ეს ხშირ შემთხვევაში იმის ბრალია, რომ არასანდო საიტს მიეცით უფლება, რომ გამოგიგზავნოთ ქუქიები, რომლებიც +18 ფოტოებს ყრიან ეკრანზე.თქვენ ქუქიების გაწმენდა შეგიძლიათ ნებისმიერ ბრაუზერში შესვლით და მათი ქეშის(Cache) გაწმენდით.

* რამდენნაირი საიტი არსებობს?

  არსებობს საიტები, რომლებსაც წერენ კოდერები და არსებობენ საიტები, რომლებსაც აწყობენ ძრავებზე(მაგალითად wordpress ძრავი).საქმე ისაა, რომ ვორდპრეს საიტები უფრო ბიუჯეტურია და რა თქმა უნდა მათი ექსპლოიტირება უფრო მარტივია, რადგან wordpress ძრავს ბევრი ბაგი(bug) ანუ მოწყვლადობა აქვს.Metasploit frameworkს უამრავი ექსპლოიტი აქვს ვორდპრესთან დაკავშირებით.

ახლა ვისაუბროთ საიტების მთავარ ფაილებზე/ფოლდერებზე.მთავარი html ფაილი არის index.html, მთავარი css ფაილი არის index.css, ხოლო მთავარი javascript ფაილი არის intex.js და არის რაღაც ფაილები, რომლებსაც ახლავე ვახსენებთ.არსებობს ფაილი robots.txt სადაც წერია ისეთი ლოკაციები, სადაც ჩვეულებრივი კლიენტი არ უნდა შევიდეს(წვდომა არ უნდა მიეცეს).აი მაგალითად ესაა googleის robots.txt ფაილის არასრული სქრინშოტი(კიდევ ბევრი რამ ეწერა უბრალოდ აღარ დავსქრინეთ):



დააკვირდით და ნახავთ, რომ ზოგ ლოკაციას უწერია Allow(დაშვება), ხოლო ზოგს Disallow(არ დაშვება).ეს უკვე უსაფრთხოების საკითხებია.

ასევე არსებობს ფოლდერი ვებსაიტებზე "/cgi-bin".CGI ნიშნავს Common Gateway Interface.ის ძალიან საჭიროა საიტის ვიზუალისთვის, მასში ყრია სკრიპტები, რომლებიც სერვერს სჭირდება.არსებობს ამის ექსპლოიტებიც.ამით იმის თქმა გვინდა, რომ ყველაფერი მოწყვლადია რაღაცის მიმართ.

ახლა ალბათ გაინტერესებთ არის კიდევ თუ არა რაიმე ენა ვებში.პასუხი არის კი!ეს გახლავთ PHP("ვარდისფერი სპილოოო" – D1sM3 1876 წელი).PHP არის Back-End ენა, რომელიც ეხმარება სერვერებს და მონაცემთა ბაზებს(Data base) ერთმანეთთან კომუნიკაციაში.PHPის ტეგი საკმაოდ უცნაურია htmlში.მისი დაგი ასე იწერება: <?php ?>

თავი 16
Introduction to Firewall

მგონი გსმენიათ ფაერვოლზე(firewall).ის არის თავდაცვის სისტემა.ფაერვოლი ბლოკავს(ან პირიქით) რაღაც სერვისებს.რეალურად ეს თქვენზეა დამოკიდებული თუ როგორ დააკონფიგურირეთ თქვენს ფაერვოლს.განვიხილოთ ასეთი შემთხვევა:

დავუშვათ ფაერვოლზე დაბლოკილია გარე ქსელში ნებისმიერი HTTP და HTTPS პორტები.ეს იმას ნიშნავს, რომ ვინც ამ როუტერს დაუკავშირდება ვებსაიტებზე წვდომას ვერ აიღებს, რადგან ფაერვოლი ბლოკავს პირდაპირ(direct) კავშირს ამ პორტებზე.ამის გადალახვა(bypass) შესაძლებელია პროქსი სერვერით ან VPNით.ასევე ფაერვოლზე შეგვიძლია დავბლოკოთ კონკრეტული IP ან კონკრეტული ვებსაიტი.ამის შემდეგ ვერავინ ვერ შევა იმ კონკრეტულ ვებსაიტზე ან ვერავინ დაუკავშირდება იმ IPს, რომელიც ფაერვოლზე დავბლოკეთ.

სხვათაშორის ტროიანებს შეუძლიათ ფაერვოლს გვერდი აუარონ თუ ისინი სწორად არის დაკრაფტული.

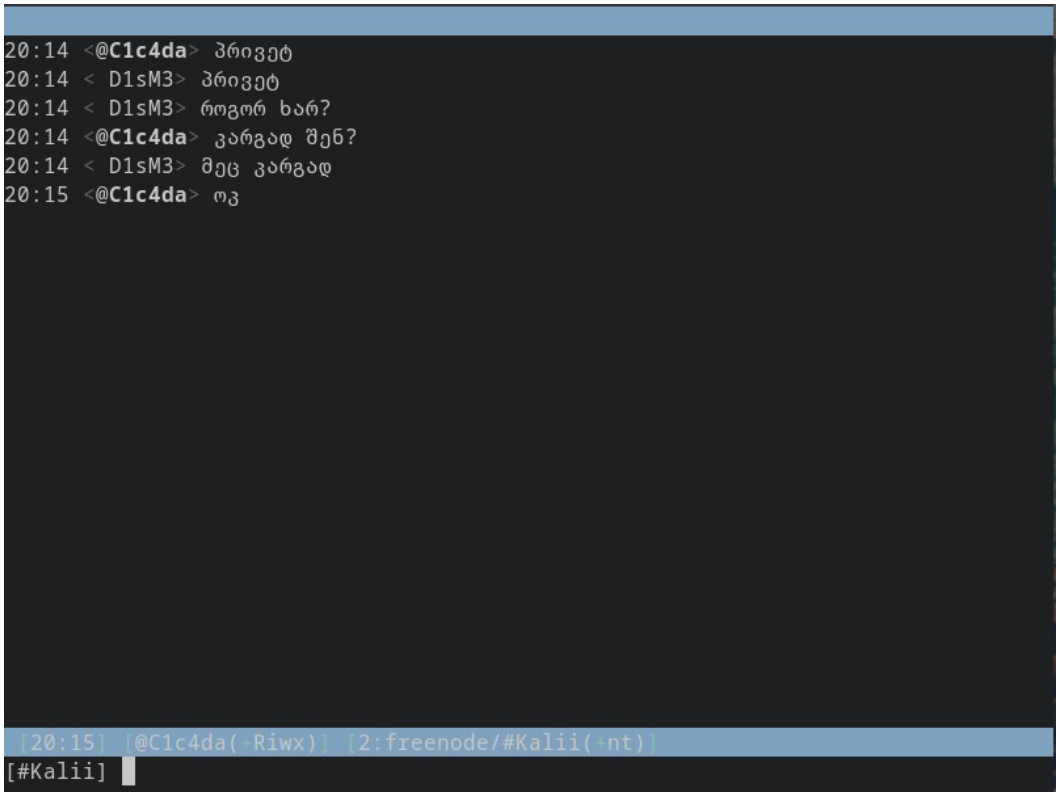
თავი 17
როგორ შეიძლება ჰაკერებმა უსაფრთხოდ იკონტაქტონ ერთმანეთში?

ახლავე გეტყვით, რომ არც მესენჯერი და არც ტელეგრამი არ გიშველით.საუკეთესო გზა ორნაირია: ან უნდა იყოს ჰაკერული ფორუმი(საიტი) და იქ მიწეროთ ერთმანეთს, ან უნდა გქონდეთ IRC ჩატ სერვერი.

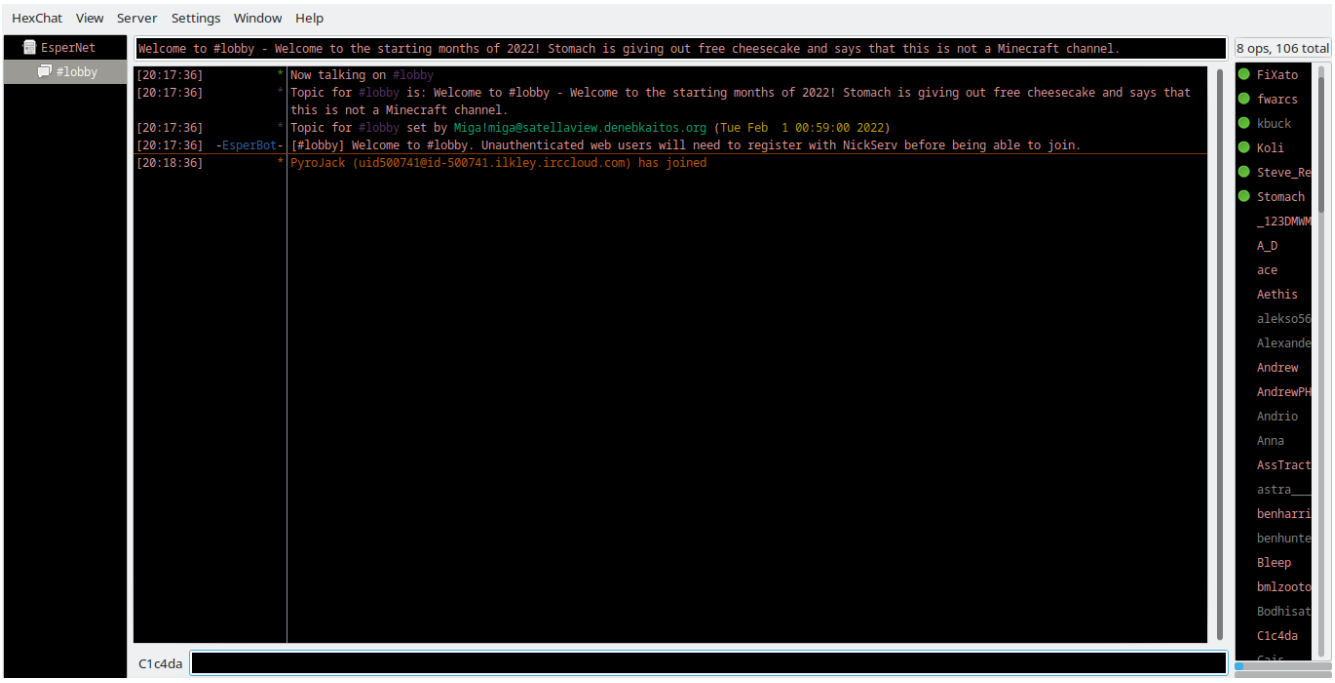
* რა არის IRC?
  IRC(Internet relay chat) არის პროტოკოლი, რომლის მეშვეობითაც ხდება ადამიანებს შორის კონტაქტი.

* ოდესმე ვინმე იყენებდა მას?
  რადგან შექმნილია ანუ იყენებდნენ.საქართველოში ეს დიდად არაა გავრცელებული და არც არასოდეს ყოფილა, მაგრამ მაგალითად შტატებში 90იანების დასრულების შემდეგ IRC ძალიან პოპულარული გახდა.

გვინდა გაიაზროთ, რომ IRC არ არის კონკრეტული ჩატ აპლიკაცია.ის არის პროტოკოლი.აქედან გამომდინარე არსებობს უამრავი IRC ჩატ კლიენტი მაგალითად irssi:



არის ისეთი კლიენტები, რომლებსაც GUI აქვთ.მაგალითად HexChat:



IRC თემაში გარკვევას დრო უნდა.იმიტომ, რომ აქ არსებობენ ჩვეულებრივი კლიენტები, ბოტები, მოდერატორები, ადმინისტრატორები და ასე შემდეგ.ამ ეკოსისტემას რომ გაუგოთ ცოტა დრო დაგჭირდებათ, მაგრამ ბოლოს სასიამოვნო ხდება.თქვენ თავისუფლად შეგიძლიათ საკუთარი IRC სერვერი დაჰოსტოთ და თუ რამდენიმე ადამიანი ხართ შეკრებილი და ერთად ჰაკერული გჯუფის შექმნას ცდილობთ, საკუთარი ჩატ სერვერი, რომელსაც როგორც გინდათ ისე დააკონფიგურირებთ არ გაწყენდათ.

ჩვენ ვთვლით, რომ საუკეთესო გზა კონტაქტისთვის Advanced IRC სერვერის ქონაა, მაგრამ თუ თქვენ ცოდნა ხელს გიწყობთ, შექმენით საიტი(ფორუმი) სადაც IT თემატიკით დაინტერესებული ადამიანები იქნებიან დარეგისტრირებულები.

მაგალითისათვის გეტყვით, რომ ჩვენ ვამუშავებთ 2 არაოფიციალურ საიტს(ჰაკერულ ფორუმს) და 3 IRC ჩატ სერვერს სადაც ხალხი ერთმანეთს ცოდნას უზიარებს და ერთად მუშაობენ.

თავი 18
Payloads && Reverse Shells

რა არის ფეილოუდი(payload)?ფეილოუდი არის კოდი, რომელსაც თავდამსხმელი სამიზნე მოწყობილობაში აინჯექტირებს.შემდეგ ეს ფეილოუდი თავდამსხმელისთვის რაიმეს აკეთებს.მაგალითად ხსნის პორტს, რათა შემდეგ

თავდამსხმელი დაუკავშირდეს ამ პორტს და შეძლოს სამიზნის shellზე წვდომის აღება.ამას ეწოდება reverse shell.ჩვენ მარტივად შეგვიძლია reverse shellის შექმნა:

ncat ბრძანება მოყვება nmapს(პორტ სკანერს, რომელიც განვიხილეთ ზევით, მაგრამ ქვედა თავშიც განვიხილათ დაწვრილებით).ეს ბრძანება საშუალებას იძლევა დავუკავშირდეთ სხვადასხვა მოწყობილობებს სხვადასხვა პორტებზე ან ჩვენთვითონ გავხსნათ პორტი.განვიხილოთ ბრძანება, რადგან ესაა ტიპიური reverse shell:

1) ncat – ეს არის თვითონ ბრძანება.
2) -lv – ეს არის ბრძანების დროშა(flag), რომელსაც ქვედა თავებში განვიხილავთ.
3) 1234 – ეს რიცხვი არის პორტის რიცხვი.ანუ რომელი პორტი უნდა გაიღოს.
4) -e – არის კიდევ ერთი ncat ბრძანების დროშა, რომლის დაწერის შემდეგაც საშუალება გვეძლევა რაიმე executable(პროგრამა) დავაყოლოთ პორტს(ანუ კლიენტი რომ პორტზე შემოვა, იმ პროგრამას გამოიყენებს).
5) /bin/bash – ეს არის ფაილი, რომელიც გდია /bin ფოლდერში და არის bash shell(ნიჟარები 7.1 თავშია განხილული).

ეს ბრძანება რამდენი ხანიც იქნება ჩართული, იმდენი ხანი იქნება იმის საფრთხე, რომ შიდა ქსელიდან ვინმე დაუკავშირდება ჩემს IP მისამართს ამ პორტზე და ჩემს კომპიუტერს ბრძანებებს გადასცემს.ჩვენ ეს ბრძანება შეგვიძლია ფაილში ჩავიტანოთ და ეს ფაილი(ანუ payload) სადმე ისეთ ადგილას დავმალოთ, სადაც მომხმარებელი ვერ იპოვის, მაგალითად /etc ფოლდერში.შემდეგ autorunზე მივათითებინოთ, რომ გაუშვას ეს ფაილი და საბოლოოდ მივიღებთ ასეთ შედეგს: სამიზნე კომპიუტერი როდესაც ჩაირთვება, ეს პროგრამა მაშინვე გაიღვიძებს და გახსნის 1234 პორტს სამიზნის კომპიუტერზე, რათა შემდეგ თავდამსხმელი დაუკავშირდეს და წვდომა აიღოს კომპიუტერზე.

თავი 19
ლინუქს ტერმინალის გამოყენება

ლინუქსის ტერმინალის გამოყენება საკმაოდ ადვილია.თქვენ უბრალოდ ბრძანებები და მათი დროშები უნდა დაიმახსოვროთ.ჩვენ ამ თავში მოგცემთ საბაზისო ცოდნას ლინუქსის ტერმინალთან მუშაობის შესახებ.ჯერ უნდა გაიაზროთ ის, რომ ლინუქსის ტერმინალს აქვს სამუშაო ფოლდერი(Working folder).ანუ მაგალითად თუ თქვენ ტერმინალში დაწერთ ბრძანებას:

pwd

და დააწვებით “Enter”ს, output გაჩვენებთ თუ რომელ ფაილში იმყოფებით ახლა(სამუშაო ფაილში).მაგალითად თუ თქვენ ხართ /home/user/Desktop ფოლდერში, ეს იმას ნიშნავს, რომ თქვენი სამუშაო ფოლდერია დესკტოპი.

მოდით ახლა ვისწავლოთ, თუ როგორ უნდა გავიგოთ რა ფაილები და ფოლდერები ყრია ჩვენს სამუშაო ფოლდერში.ამისათვის საჭიროა, რომ დაწეროთ ბრძანება:

ls

ბრძანების გაშვების შემდეგ(ანუ “Enter”ს რომ დააწვებით), ის გიჩვენებთ ყველა იმ ფოლდერსა თუ ფაილს, რომელიც თქვენს სამუშაო ფოლდერშია(გარდა დამალული ფაილებისა და ფოლდერებისა, რომლებსაც მალე შევეხებით).აი მაგალითი:

როგორც ხედავთ, ვიმყოფებით “/home/cicada” ფოლდერში და როდესაც დავწერეთ ბრძანება “ls” მან ყველა ის ფოლდერი გვიჩვენა, რომლებიც სამუშაო ფოლდერში ეყარა.დავუშვათ ჩვენ გვინდა სხვა ფოლდერის, მაგალითად “/dev”ში ჩახედვა და ფაილებისა და ფოლდერების შემოწმება.ამისათვის ვწერთ ბრძანებას:

ls /dev

თქვენ რომელ ფოლდერშიც არ უნდა იყოთ, ის მაინც იმ ფაილებს და ფოლდერებს გიჩვენებთ, რომლებიც “/dev” ფოლდერში ყრია.

ახლა ვისაუბროთ დამალულ ფაილებზე და ფოლდერებზე.ეს ისეთი ფაილები და ფოლდერებია, რომლებიც ძირითადად არ ჩანს “ls” ბრძანების outputში და ფაილების მენეჯერში, რადგან ისინი ძირითადად რაღაც პროგრამების config ფაილებია და მათი ნახვა არავის სურს.მაგრამ თუ თქვენ გინდათ მათში ჩახედვა და ჩასწორება(თუ ამის ცოდნა და სურვილი გაქვთ) მაშინ წერთ ბრძანებას:

ls -a

“-a” არის “ls” ბრძანების დროშა, რომელიც გვიჩვენებს ყველა ფოლდერსა და ფაილს, დამალული ფოლდერებისა და ფაილების ჩათვლით.

ახლა განვიხილოთ თუ როგორ უნდა შეიცვალოთ სამუშაო ფოლდერი.ამისათვის დაგჭირდებათ “cd”(Change dir) ბრძანება.ეს ბრძანება საშუალებას იძლევა ფოლდერში შევიდეთ ან პირიქით, გამოვიდეთ.მაგალითად: თუ თქვენ ხართ “/dev/cpu” ფოლდერში(რატომღაც) და გინდათ, რომ იქიდან გამოხვიდეთ(კარგი არჩევანია) საჭიროა დაწეროთ ბრძანება:

cd ..

“..” ნიშნავს ერთი ფოლდერით უკან გამოსვლას.თუ თქვენ ხართ მაგალითად “/dev” ფოლდერში და რატომღაც მოგინდათ “cpu” ფოლდერში გადასვლა მაშინ წერთ ბრძანებას:

cd cpu

ამის შემდეგ თქვენ აღმოჩნდებით “/dev/cpu” ფოლდერში.

დავუშვათ თქვენ გინდათ, რომ რაიმე ფაილიდან ტექსტი ამოიკითხოთ(ანუ ფაილის შიგთავსი ნახოთ).ამისათვის არსებობს cat ბრძანება.მაგალითად თქვენს სამუშაო ფოლდერში არის ფაილი “kazinosEqauntebi.txt”(მე-13 თავში ვიხუმრეთ) და გინდათ, რომ გაიგოთ რა წერია მასში.წერთ ბრძანებას:

cat kazinosEqauntebi.txt

ამის შემდეგ ტერმინალში დაიწერება მთლიანი ტექსტი, რაც კი იმ ფაილში ეწერა თუნდაც სიგრძით 1 000 000 ხაზი იყოს.

მგონი ტერმინალში basic მოქმედებები გაიაზრეთ.ახლა ავხსნათ რა არის ბრძანების დროშა.ბრძანებებს აქვთ თავიანთი არგუმენტები(გაიხსენეთ მე-8 თავში ფუნქციები რომ ავხსენით იგივე პრინციპია).მაგალითად ახლა ზემოთ, რომ დაწერეთ “cat kazinosEqauntebi.txt”, მანდ “kazinosEqauntebi.txt” ფაილიც არგუმენტია.არგუმენტები ამოიყოფა დროშებით, რადგან ყველა არგუმენტი რაღაცას აკეთებს.მოდით მე-18 თავში ნახსენები ბრძანება განვიხილოთ:

ncat -l 1234 -v -e /bin/bash

ამ ბრძანების შემდეგ რასაც კი საწყისად “-” სიმბოლო ექნება, დაიმახსოვრეთ, რომ ყველაფერი ეგეთი არის ბრძანების დროშა(არსებობენ დროშები, რომლებსაც “--” აქვთ წინ და არა”-”).ყველა დროშის შემდეგ რაღაც იწერება.მაგალითად “-l” დროშა ნიშნავს “listen”ს ანუ რომელ პორტს უნდა უსმინოს.ჩვენ მას გადავაწოდეთ არგუმენტად რიცხვი 1234(ანუ 1234-ე პორტი).შემდეგ მოდის “-v” დროშა.ამ დროშას არგუმენტი არ სჭირდება.ეს დროშა ნიშნავს “verbose”ს და რეალურად ასენსიტიურებს ბრზანების outputს(ანუ ამის გარეშე არ დაიწერებოდა ვინ დაუკავშირდა პორტს და რომელი პორტიდან).შემდეგ მოდის “-e” დროშა, რომელიც ნიშნავს “execute”ს და მას ვაწვდით არგუმენტად რომელიმე executable ფაილს(ამ შემთხვევაში bash ნიჟარას).

თუ თქვენ გინდათ, რომ კონკრეტული ბრძანების ყველა დროშა და მათი დეფინიცია გაიგოთ დაწერეთ ბრძანება:

man <აქ ის ბრძანება, რომელიც გაინტერესებთ>

ამან თუ არ იმუშავა, მაშინ უბრალოდ დაწერეთ:

<აქ ის ბრძანება, რომელიც გაინტერესებთ> --help

იმედია ყველაფერი ნათელია.

თავი 20
Nmap

ჩვენ ამ წიგნში ქსელიც და ინტერნეტიც განვიხილეთ(ზედაპირულად, მაგრამ მაინც).”nmap” ბრძანება ასკანირებს remote მოწყობილობას და გვეუბნება თუ რა პორტები არის მასზე გახსნილი.Default scan მასზე არის TCP პორტების სკანირება.მაგრამ არის დროშები, რომლებითაც შეგვიძლია მას UDP პორტების სკანირება ვაიძულოთ(ამას root პრივილეგიები სჭირდება).მოდით მაგალითებისათვის ავიღოთ IP “11.22.33.44”.ჩვეულებრივი პორტ სკანირება(1დან 1000ის ჩათვლით პორტის სკანირება) ასეთი იქნება:

nmap 11.22.33.44

ეს basic ინფორმაციას მოგვცემს remote მოწყობილობაზე, მაგრამ არაფერს ისეთს, რაც მის ექსპლოიტირებაში დაგვეხმარება.თუ თქვენ გინდათ, რომ რომელიმე სპეციფიური პორტი დაასკანიროთ(მაგალითად SSH ანუ 22-ე პორტი), წერთ ასეთ რამეს:

nmap -p 22 11.22.33.44

ის მხოლოდ იმ პორტს დაასკანირებს, რომელსაც ეტყვით.ჩვენ თუ გვინდა უფრო დეტალური ინფო მივიღოთ სამიზნეზე ვწერთ ასეთ რამეს:

sudo nmap -A 11.22.33.44

ეს ბრძანება უკვე root პრივილეგიებს მოითხოვდა, ამიტომაც დავწერეთ წინ “sudo”(“sudo ჯადოსნური სიტყვაა” – D1sM3 1121 წელი დიდგორი).თუ მაგალითად თქვენ გინდათ, რომ პორტების სერვისებზე მიიღოთ დეტალური ინფორმაცია, მაშინ წერთ ასეთ რამეს:

sudo nmap -A -sC -sV 11.22.33.44

ეს ყველაფერს გეტყვით სამიზნეზე და მის სერვისებზე.თუ მაგალითად სამიზნეს გახსნილი აქვს HTTP პორტი(მაგალითად apache2ით), ის გეტყვით apache2ის რა ვერსია აყენია სამიზნეზე და ამის შემდეგ თქვენ შეძლებთ უკვე იპოვოთ საჭირო ვერსიის ექსპლოიტი და დააექსპლოიტოთ ის(ძველი apache2ის ვერსიებზე მუშაობს Log4Shell ექსპლოიტი, რადგან ის ჯავას logging ბიბლიოთეკას, Log4jს იყენებდა).

დასასრული
:(

წიგნს აქ ვამთავრებთ, რადგან 200 საუკუნიანი მსჯელობის შემდეგ გადაწყდა, რომ წიგნში ძალიან advanced თემებს ვერ შევეხებით.სამაგიეროდ ჩვენ მოგცემთ მოტივაციას.ეს არის ლინკი, საიდანაც შეგიძლიათ გადმოიწეროთ cisco packet tracer(პროგრამა, სადაც შეგიძლიათ real life ქსელის სიმულაციები შექმნათ).

Link for Cisco Packet Tracer:
https://www.packettracernetwork.com/download/download-packet-tracer.html

საბოლოოდ იმას გეტყვით, რომ ძალიან გთხოვთ ნუ გადაყრით ქართულ “აიტი აკადემიებში” ფულს სადაც ასწავლიან “ეთიკურ ჰაკინგს”.ეს ყველაფერი უბრალოდ ცირკია, რადგან არ არსებობს ადამიანი, რომელსაც შეუძლია სხვას ჰაკინგი ასწავლოს.ჰაკინგი ხელოვნებაა, რომელიც შენთვითონ უნდა შეიგრძნო.თუ მას ვერ გრძნობ, გამოდის რომ ვერასოდეს შეძლებ.ესაა მწარე რეალობა, რომელსაც არავინ არასოდეს გეტყვით <3