ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ

«Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ» КЕАҚ

Студенттер мен жас ғалымдардың
«ǴYLYM JÁNE BILIM - 2025»
XIX Халықаралық ғылыми конференциясының
БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ
XX Международной научной конференции
студентов и молодых ученых
«ǴYLYM JÁNE BILIM - 2025»

PROCEEDINGS
of the XX International Scientific Conference
for students and young scholars
«ǴYLYM JÁNE BILIM - 2025»

2025
Астана

УДК 001(06)
ББК 72я631
F96

«ǴYLYM JÁNE BILIM – 2025» студенттер мен жас ғалымдардың XX Халықаралық ғылыми конференциясы = XX Международная научная конференция студентов и молодых ученых «ǴYLYM JÁNE BILIM – 2025» = The XX International Scientific Conference for students and young scholars «ǴYLYM JÁNE BILIM – 2025». – Астана: – 3813 б. - қазақша, орысша, ағылшынша.

ISBN 978-601-08-5373-7

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.В сборник вошли доклады студентов, магистрантов, докторантов имолодыхученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 001(06)
ББК 72я431
F96

ISBN 978-601-08-5373-7

Л.Н. Гумилев атындағы Еуразия ұлттық университеті, 2025

СЕКЦИЯ 2

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И КОММ УНИКАЦИОННЫЕ ТЕХНОЛОГИИ

Подсекция 2.2

Интеллектуальные информационные системы

255.	Сафонова Софья Александровна «Современные аспекты информационной безопасности в облачных вычислениях: модели, угрозы и методы защиты»	1034
256.	Смаилова Назгүл Батырбекқызы «Терең оқыту арқылы кітап ұсыныстарын әзірлеу: collaborative filtering, content-based және nlp әдістерінің комбинациясы»	1041
257.	Тажібай Аружан Айдосқызы, Кудубаева Сауле Альжановна «Көру қабілеті әлсіз адамдарға арналған аі дауыстық көмекші: нақты уақытта объектілерді анықтау және қашықтықты бағалау»	1046
258.	Тайжанов Азамат Жанкелдіұлы «Python тілінде фильмдердің интеллектуалды ұсыныс жүйесін әзірлеу»	1051
259.	Умирзахов Сундетали Кабылбекович «Сұраныстарды интеллектуалды талдау негізінде ұйымның сайты үшін чат-бот құру»	1055
260.	Шайхстан Марғұлан «IOT Сенсорлары негізінде ауа ластану деңгейін болжау»	1060

Подсекция 2.3

Современные тенденции в программной инженерии и управлении в условиях цифровой индустрии

261.	Bekenova A.B. «Development of a registration panel for users and doctors with integration into the database»	1077
262.	Bolat A.Zh. «Data analysis methods and decision making using big data and machine learning tools»	1081
263.	Алтайұлы А. «Visual studio интегралды ортасында «қойма қызметкерлеріне арналған» мәліметтер қорын жобалау»	1086
264.	Арап А.Қ. «Ақылды сурет салушы роботты әзірлеу»	1088
265.	Артыкбекқызы А. «Ақылды үйлердегі заттар интернеті(iot) мен робототехниканың өзара әрекеттесуі»	1091
266.	Ахметова А.Д. «Тоңазытқыштағы өнімдерді бақылауға және тағам әзірлеу ұсынысын беруге арналған программалық қосымша»	1096
267.	Дәрібай Д.Д. «Робототехниканы қолдану арқылы қойма логистикасындағы қолданыстағы басқару жүйелерін талдау»	1100
268.	Жамбулов С.Ж. «Білім алушыларды информатика және программалау олимпиадаларына дайындауда жасанды интеллекттің қолданысы»	1102
269.	Каиржан Р.С. «Development of system for recognition of emotional states of employees based on computer vision methods on Raspberry Pi»	1108
270.	Кайрекенова Н.Р. «Өнеркәсіптік роботты көру үшін машиналық оқытудың заманауи тәсілдері: әдістер, деректер жиынтығы және оптимизациялау»	1111

СЕКЦИЯ 3
ЕСТЕСТВЕННЫЕ НАУКИ

УДК 004.056.5

ELF ТАЛДАУЫНДАҒЫ CAPSTONE: СЫЗЫҚТЫҚ ЖӘНЕ РЕКУРСИВТІ ДИЗАССЕМБЛЕРЛЕУ

Кеттеш Бекнұр Нұрланұлы
kbeknurr@mail.ru
Л.Н.Гумилев атындағы ЕҰУ «Ақпараттық технологиялар» факультетінің 2-курс магистранты, Астана, Қазақстан
Ғылыми жетекшісі – Ж.Сауханова

Дизассемблерлеу – екілік файлдарды талдаудың негіздерінің бірі. Дизассемблерлеу машина кодын неғұрлым түсінікті ассемблер көрінісіне түрлендіруге және бағдарламаның құрылымы мен логикасын тереңірек түсінуге мүмкіндік береді. Бағдарламалық кодты қорғау мен обфускациялаудың үнемі күрделене түсуіне байланысты мақалада ұсынылған әдістерді қолдану нәтижесінде талдаушы өзінің зерттеуінің дәлдігі мен тиімділігін едәуір арттыра алады.

Бинарлық файлдарды талдауда дизассемблердің стандартты болжамдары бұзылған жағдайда дизассемблердің арнайы өтуі қажет болады. Мысалы, зиянды бағдарламаны, обфускацияланған файлдарды, жады немесе бағдарламалық жасақтаманың дамптарын талдау барысында. Сонымен қатар, кодтағы осалдықтарға нұсқайтын үлгілерді іздеу барысында да дизассемблердің арнайы өтулері бинарлық талдаудың ерекше түрлерін оңай жүзеге асыруға мүмкіндік береді. Бұл сондай-ақ дизассемблирлеудің жаңашыл әдістерімен тәжірибе жасауға мүмкіндік беретін пайдалы зерттеу әдісі.

Арнайы дизассемблирлеудің алғашқы нақты мысалы ретінде командалардың қабаттасуы (overlapping instructions) арқылы код обфускациясын зерттеу қарастырылады. Көптеген дизассемблерлер әрбір бинарлық файл үшін тек бір листинг жасайды, өйткені олар әрбір байт тек бір ғана командаға қатысты екенін білдіреді, әрбір команда бір қарапайым блокқа тиесілі және әрбір қарапайым блок бір ғана функцияның бөлігі болып табылады деп болжайды. Басқаша айтқанда, дизассемблерлер кодтың бөліктері бір-бірімен қабаттаспайды деп есептейді. Алайда, командалардың қабаттасуы бұл болжамды бұзады, кері инженерияны қиындатады. Командалардың қабаттасуы x86 платформасында мүмкін, өйткені бұл платформада командалардың ұзындығы әртүрлі (ARM сияқты кейбір басқа платформалардан айырмашылығы, онда барлық командалар бірдей байт санынан тұрады). Сондықтан процессор командаларды жадыда арнайы туралауды талап етпейді, бұл бір команданың басқа команда алған мекенжайларды иеленуіне мүмкіндік береді. Осыған байланысты x86 жүйесінде дизассемблирлеуді команданың ортасынан бастап, алғашқы командамен ішінара немесе толықтай қабаттасатын басқа команданы алуға болады.

Capstone – бұл x86/x86-64, ARM, MIPS және басқа да ең танымал командалар архитектураларын өңдеуге мүмкіндік беретін қарапайым әрі жеңіл API ұсынатын дизассемблирлеу кітапханасы. Ол C/C++ және Python (сондай-ақ басқа да тілдер) интерфейстеріне ие. Кітапхана Windows, Linux және macOS сияқты барлық танымал платформаларда жұмыс істейді. Capstone мүлдем тегін және ашық бастапқы кодқа ие.

Capstone көмегімен дизассемблирлеу құралдарын жасау – өте икемді әрі қарапайым процесс. API негізінде бірнеше ғана функциялар мен деректер құрылымдары болғанымен, бұл оны қолданудың ыңғайлылығына еш әсер етпейді. Capstone дизассемблирленген командалар туралы маңызды мәліметтерді алуды айтарлықтай жеңілдетеді. Оларға операция кодтары, мнемоникалар, класс, команда пайдаланатын және өзгеретін регистрлер мазмұндары және тағы басқа деректер жатады.

Зерттеу барысында дизассемблерлеу процестерін тереңірек талдауға мүмкіндік беретін Capstone кітапханасын қолдану арқылы екі түрлі әдіс қарастырылды. Бұл екі тәсіл – сызықтық дизассемблерлеу және рекурсивті дизассемблерлеу – бір қарағанда ұқсас болғанымен, олардың жұмыс істеу принциптері мен қолдану салалары айтарлықтай ерекшеленеді.

Алдымен сызықтық дизассемблерлеу мысалы қарастырылады.

Python көмегімен x86_64 платформасы үшін Сурет 1-де дизассемблерлеу сызықты дизассемблерлеудің мысалы келтірілген:

```
# -*- coding: utf-8 -*-

from capstone import *

CODE = b"\x55\x48\x8b\x05\xb8\x13\x00\x00"

md = Cs(CS_ARCH_X86, CS_MODE_64)
for i in md.disasm(CODE, 0x1000):
    print("0x%x:\t%s\t%s" % (i.address, i.mnemonic, i.op_str))
```

Сурет 1. linear_disassembler.py файлы

Бұл код Capstone көмегімен бинарлық кодты x86_64 ассемблер тіліне аударады. Енді әр жолды егжей-тегжейлі қарастырайық.

1. Capstone кітапханасын импорттау

from capstone import *

Бұл жол Capstone кітапханасын жүктейді, осылайша оны Python кодымызда қолдана аламыз.

2. Бинарлық кодты анықтау

CODE = b"\x55\x48\x8b\x05\xb8\x13\x00\x00"

Бұл x86_64 платформасында орындалатын бинарлық код. Әрбір \x.. мәні – бұл машиналық кодтың бір байты.

3. Дизассемблер объектісін жасау

md = Cs(CS_ARCH_X86, CS_MODE_64)

Мұнда Capstone дизассемблер объектісін құрылады. Ол екі негізгі параметрді алады:

CS_ARCH_X86 – x86 архитектурасын көрсетеді

CS_MODE_64 – 64-биттік режимде жұмыс істеу керектігін білдіреді

4. Дизассемблирлеу процесі

for i in md.disasm(CODE, 0x1000):

    print("0x%x:\t%s\t%s" % (i.address, i.mnemonic, i.op_str))

Бұл цикл Capstone дизассемблерін CODE айнымалысына қолданады және оның нәтижесін экранға шығарады. 0x1000 – виртуалды мекенжай, яғни дизассемблерленген кодтың бастапқы орны ретінде қабылданатын мекенжай.

Әрбір нұсқау үш негізгі өріске бөлінеді:

i.address – команда орналасқан мекенжай

i.mnemonic – команданың ассемблерлік мнемоникасы

i.op_str – команданың операндтары

linear_disassembler.py файлын іске қосу нәтижесі Сурет 2-де көрсетілген.

```
binary@binary-VirtualBox:~$ python linear_dissasembler.py
0x1000: push    rbp
0x1001: mov     rax, qword ptr [rip + 0x13b8]
```

Сурет 2. linear_disassembler.py кодының нәтижесі

Нәтиже төмендегідей болады:

0x1000: push rbp

0x1001: mov rax, qword ptr [rip + 0x13b8]

Бұл бинарлық кодты оқылатын ассемблер тіліне айналдырғанымызды білдіреді. Әр жолды жеке қарастырайық:

0x1000: push rbp

push rbp командасы стекке rbp регистрінің мәнін сақтайды.

Бұл функцияның прологы ретінде жиі қолданылады. Келесі жол:

0x1001: mov rax, qword ptr [rip + 0x13b8]

mov rax, qword ptr [rip + 0x13b8] – rip регистріндегі мәнге 0x13b8 санын қосып пайда болған мәнге сәйкес адресте орналасқан 8 байтты деректі rax регистріне жүктейді.

Мұндай операция динамикалық жүктелетін деректерді немесе жадыдағы мәндерді оқу үшін қолданылады.

Сызықтық дизассемблер кодты біртіндеп, рет-ретімен талдайды және оның ішінде қандай да бір тармақталу немесе секіру орын алса да, ол келесі команданы бірізді түрде оқи береді. Мұнда Capstone кітапханасының CS класын қолданылды, ол x86_64 архитектурасында 64-биттік кодты дизассемблерлеуге мүмкіндік береді. Алдымен машиналық код CODE айнымалысына бинарлық түрде енгізіліп, содан кейін ол md.disasm() функциясы арқылы талданды. Нәтижесінде дизассемблерленген кодтың түсінікті форматы алынғанын көруге болады. Дегенмен, бұл тәсілде кодтың орындалу бағыты өзгерсе немесе секіру командалары қолданылса, талдау толық болмай қалуы мүмкін. Сондықтан, күрделі бағдарламаларды зерттеу үшін рекурсивті дизассемблерлеу әдісін пайдалану қажет.

Рекурсивті дизассемблер кодтың орындалу бағытын ескере отырып, дизассемблерлеу жасайды. Сурет 3-те көрсетілген recursive_disassm() функциясы белгілі бір мекенжайдан бастап дизассемблерлеуді жүзеге асыратындай етіп құрылды. Бұл әдіс кодтың ішінде кездесетін jmp және call командаларын талдайды және олар сілтейтін жаңа мекенжайларды да дизассемблерлеу үшін queue тізіміне қосады. Осылайша, орындалу барысында шын мәнінде қандай командалар орындалатыны нақты анықталады.

```
# -*- coding: utf-8 -*-

from capstone import *

def recursive_disasm(code, start_addr):
    visited = set()
    queue = [start_addr]
    md = Cs(CS_ARCH_X86, CS_MODE_64)

    while queue:
        addr = queue.pop()
        if addr in visited:
            continue
        visited.add(addr)

        for i in md.disasm(code, addr):
            print("0x%x:\t%s\t%s" % (i.address, i.mnemonic, i.op_str))
            if i.mnemonic in ['jmp', 'call']:
                try:
                    queue.append(int(i.op_str, 16))
                except ValueError:
                    pass  # Если переход не на конкретный адрес

CODE = b"\x55\x48\x8b\x05\xb8\x13\x00\x00\xff\xe0"  # содержит "jmp rax"
recursive_disasm(CODE, 0x1000)
```

Сурет 3. recursive_disassembler_code файлы

Нәтижесінде, егер кодта jmp немесе call кездессе, рекурсивті дизассемблер олардың көрсеткен жаңа мекенжайларын да талдап, толық талдау жүргізеді. Тәжірибе көрсеткендей,

бұл әдіс сызықтық дизассемблерлеуге қарағанда анағұрлым тиімді және сенімді. Мысалы, jmp rax командасы орындалған кезде, келесі орындалатын команда жаңа мекенжайда болуы мүмкін, ал рекурсивті дизассемблер осы жағдайды ескереді. Обфускаторлар ешқашан пайдаланылмайтын жалған жолдарды құру арқылы көбінесе сызықтық дизассемблерлерді шатастыруға тырысады. Ол үшін әрдайым ақиқат немесе әрдайым жалған болатын предикаттар қолданылады, бірақ дизассемблер оларды тани алмайды. Мұндай қиын анықталатын предикаттар көбінесе сандық теңдіктерді немесе көрсеткіштерді біріктіру әдістерін қолдану арқылы жасалады.

Кодта кездесетін jmp командасының талдауы Сурет 4-те көрсетілген.

```
binary@binary-VirtualBox:~$ python recursive_disassemler.py
0x1000: push    rbp
0x1001: mov     rax, qword ptr [rip + 0x13b8]
0x1008: jmp     rax
```

Сурет 4. recursive_disassembler кодының нәтижесі

Неліктен арнайы дизассемблер үзіндісін жазу керек?

IDA Pro, Ghidra, Radare2 сияқты стандартты дизассемблерлер кең мүмкіндіктерге ие, бірақ әрқашан кодты дұрыс түсіндіре бермейді. Проблемалар келесі жағдайларда пайда болады:

- Кодты бұзу. Кейбір екілік файлдарда артық нұсқаулар, өзін-өзі өзгертетін код бар, бұл талдауды қиындатады.
- Стандартты емес нұсқауларды пайдалану. Мысалы, Кеңейтілген командалар жиынтығы бар процессорларға (SSE, AVX) стандартты құралдар қолдау көрсетпеуі мүмкін.
- Код пен деректерді бөлу. Кейбір ELF бөлімдерінде дизасемблер орындалатын код ретінде қате түсіндіретін деректер болуы мүмкін.

Қорытынды

Зерттеу барысында Capstone кітапханасының көмегімен бинарлық кодты дизассемблерлеу әдістерінің екі түрі – сызықтық және рекурсивті тәсілдері талданды. Сызықтық дизассемблерлеу қарапайым, тікелей және жылдам болып табылады, алайда тармақталу және секірулер бар жағдайларда кейбір маңызды нұсқауларды жіберіп алуы мүмкін. Ал рекурсивті тәсіл орындалу логикасын толығырақ ескере отырып, кодтың барлық ықтимал орындалу жолдарын зерттеуге мүмкіндік береді, бұл күрделі бағдарламаларды талдауда аса маңызды.

Қорытындылай келе, әрбір әдістің өз артықшылықтары мен кемшіліктері бар екенін атап өткен жөн. Сызықтық тәсіл қарапайым және жылдам талдау жүргізуге мүмкіндік берсе, рекурсивті тәсіл кодтың нақты орындалу ағымын толық түсінуге септігін тигізеді. Capstone кітапханасының икемді API-і бұл екі әдісті де тиімді жүзеге асыруға мүмкіндік беріп, бинарлық файлдарды талдау, кері инженерия және қауіпсіздік зерттеулері салаларында қуатты құрал ретінде қолданылуын қамтамасыз етеді. Осылайша, зерттеушілер өздерінің нақты қажеттіліктеріне байланысты сәйкес әдісті таңдап, талдау процесін барынша тиімді жүргізе алады.

Қолданылған әдебиеттер тізімі

1. Эндриесс Д. Практический анализ двоичных файлов. – М.: Бином, 2022. – 223 с.
2. Эриксон Д. Hacking: The Art of Exploitation. – San Francisco: No Starch Press, 2004. – 480 с.

3. Данг Б., Газет А., Бачаалани Е., Жоссе С. Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation. – Indianapolis: Wiley, 2014. – 120 с.
4. Юричев Д. Reverse Engineering for Beginners. – М.: O'Reilly, 2011. – 350 с.
5. Эйлам Э. Reversing: Secrets of Reverse Engineering. – Indianapolis: Wiley, 2005. – 78 с.

УДК 004

LINUX ҚОРҒАУДЫҢ ЗАМАНАУИ ӘДІСТЕРІНЕ ТАЛДАУ.OPENVAS ЖӘНЕ NMAP КӨМЕГІМЕН ОСАЛДЫҚТАРДЫ АНЫҚТАУ.

Көшкінбаева Фариза Қонайқызы
fari_01-21@list.ru
Қ.Жұбанов атындағы Ақтөбе өңірлік университеті, магистрант, Ақтөбе, Қазақстан
Ғылыми жетекшісі – Г.А.Шангытбаева

Аннотация. *Мақалада Openvas және Nmap құралдары арқылы осалдықтарды анықтауға ерекше назар аудара отырып, Linux операциялық жүйесін қорғаудың заманауи әдістері қарастырылады. Зерттеу желілік трафиктің осалдықтарын сканерлеу процесін, сондай-ақ Linux жүйелерінің қауіпсіздігі контекстінде осы құралдарды қолдану тиімділігін бағалауды сипаттайды. Талдау жүргізу үшін Kali Linux қолданылады — ең танымал қауіпсіздік тестілеу платформаларының бірі. Мақалада Openvas және Nmap жұмысының негізгі принциптері, олардың осалдықтарды анықтаудағы ерекшеліктері мен артықшылықтары, сондай-ақ инфрақұрылымның жалпы қауіпсіздігін жақсартуға ықпал ететін жүйелерді бақылау және шабуылдан қорғау процесіне біріктіру мүмкіндіктері егжей-тегжейлі қарастырылады.*

Кілт сөздер: *қауіпсіздік құралдары, OpenVAS, Nmap, осалдықтар, сканерлеу*

Аннотация. *В статье рассматриваются современные методы защиты операционной системы Linux, с особым акцентом на обнаружение уязвимостей с помощью инструментов OpenVAS и Nmap. В ходе исследования будет описан процесс сканирования уязвимостей сетевого трафика, а также оценка эффективности применения этих инструментов в контексте безопасности Linux-систем. Для проведения анализа используется Kali Linux — одна из самых популярных платформ для тестирования безопасности. В статье подробно рассмотрены основные принципы работы OpenVAS и Nmap, их особенности и преимущества в выявлении уязвимостей, а также возможности их интеграции в процесс мониторинга и защиты систем от атак, что способствует улучшению общей безопасности инфраструктуры.*

Ключевые слова: *инструменты безопасности, OpenVAS, Nmap, уязвимости, сканирование*

Annotation. *The article discusses modern methods of protecting the Linux operating system, with a special focus on vulnerability detection using OpenVAS and Nmap tools. The study will describe the process of scanning network traffic vulnerabilities, as well as evaluating the effectiveness of using these tools in the context of Linux system security. Kali Linux, one of the most popular security testing platforms, is used for the analysis. The article discusses in detail the basic principles of OpenVAS and Nmap, their features and advantages in identifying vulnerabilities, as well as the*