

П. А. Бережная¹, К. Д. Мирчевски¹

¹ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, Российская Федерация

ВСТРОЕННЫЕ СРЕДСТВА АРХИВИРОВАНИЯ ОС WINDOWS. ВОПРОСЫ БЕЗОПАСНОСТИ

Аннотация. В статье представлено экспериментальное исследование уязвимости нативного инструмента архивации операционной системы Windows – отсутствия встроенной поддержки парольной защиты. Цель работы – разработка и апробация методики создания криптостойких запароленных архивов с использованием исключительно встроенных средств ОС (PowerShell и .NET Framework) без привлечения стороннего программного обеспечения.

На тестовом стенде под управлением Windows 11 была реализована комбинация стандартной утилиты Compress-Archive для сжатия данных и симметричного шифрования AES-256 с ключом, вырабатываемым через PBKDF2 с настраиваемым числом итераций и криптографической солью. Проведён эксперимент по моделированию атаки полным перебором паролей с использованием разработанного скрипта на PowerShell.

Результаты наглядно продемонстрировали, что увеличение длины пароля и размера соли экспоненциально повышает сложность взлома: каждый дополнительный символ увеличивает пространство поиска в 10 раз (для цифр) или в 62 раза (для буквенно-цифровых паролей). Также показано, что снижение числа итераций PBKDF2 с 10 000 до 100 ускоряет перебор в 44,5 раза, что подчёркивает важность корректной настройки всех параметров безопасности. При этом отмечается, что при использовании современного специализированного оборудования скорость перебора возрастает на порядки, что требует применения длинных паролей и максимального числа итераций. Гипотезы исследования подтверждены, предложенное решение демонстрирует криптостойкость, которую теоретически можно сопоставить со сторонними архиваторами при условии соблюдения актуальных стандартов криптографии. Работа имеет практическую значимость для системных администраторов и обычных пользователей, а также потенциал для использования в учебном процессе по курсам информационной безопасности и криптографии.

Ключевые слова: Windows, архивирование, безопасность данных, PowerShell, шифрование, AES, встроенные средства.

Введение

В цифровую эпоху, где объёмы создаваемых и хранимых данных растут экспоненциально, архивирование остаётся фундаментальной операцией как для простых пользователей, так и для различных организаций. Оно решает задачи экономии дискового пространства, организации хранения и удобной передачи файлов. Немаловажной проблемой современности являются киберугрозы – от утечек персональных данных до целевых атак на крупные корпоративные сети, вследствие чего безопасность архивов перестала быть второстепенной задачей. Парольная защита сжатых данных стала стандартом де-факто, реализованным во многих популярных сторонних архиваторах, таких как WinRAR, 7-Zip и других. Эти приложения предоставляют пользователям удобный интерфейс для создания архивов, защищённых стойкими криптографическими алгоритмами. Парадоксально, но встроенные средства операционной системы Windows, являющейся самой распространённой десктопной платформой в мире, не предлагают прямой функциональности для создания запароленных архивов, что создаёт огромный пробел в безопасности для пользователей, которые полагаются на нативные инструменты системы, либо же по каким-то причинам не устанавливают дополнительное программное обеспечение (далее – ПО).

Проблема безопасности архивных форматов давно находится в поле зрения исследователей. Ещё в 2006 году в статье «Архиваторы под рентгеном» были детально проанализированы уязвимости алгоритмов шифрования в популярных сторонних архиваторах (таких как WinZip и WinRAR), однако они фокусировались исключительно на внешнем ПО [1]. Вопрос о создании архивного решения с помощью встроенных скриптовых средств ОС, криптостойкость которого была бы сопоставима с современными реализациями AES, к примеру, в том же WinRAR, оставался открытым. В работе Алексеева Е.К. и соавторов также справедливо отмечается, что большинство исследований сосредоточено на шифровании всего

Рубрика 2. Методы и системы защиты информации, информационная безопасность

дискового тома как единого целого, часто с использованием специализированного ПО или аппаратных токенов, при этом остаётся недостаточно изученным и систематизированным вопрос реализации «точечной» криптографической защиты отдельных пользовательских данных (в частности, архивов) с использованием встроенных средств ОС [2].

Объект исследования – область обеспечения информационной безопасности в ОС семейства Windows.

Предмет исследования – функциональные возможности и ограничения встроенных средств архивирования ОС Windows, а также потенциальные методы усиления их безопасности исключительно за счёт других нативных инструментов платформы для создания запаролённых архивов.

Цель исследования – разработать и апробировать на практике методику создания криптостойких запаролённых архивов с использованием исключительно встроенных средств ОС Windows, а также экспериментально оценить зависимость стойкости полученных архивов от ключевых параметров шифрования.

Литературный обзор

Актуальные руководства и техническая документация Microsoft подтверждают, что встроенная утилита Compress-Archive в Windows не поддерживает парольную защиту, перенаправляя пользователей к использованию сторонних решений, но в то же время платформа предоставляет разработчикам полный доступ к криптографическим примитивам через пространство имён System.Security.Cryptography в .NET, включая реализацию AES и PBKDF2 для выработки ключей [3] [4]. Имеется очевидный парадокс: ОС обладает всеми необходимыми криптографическими инструментами, но не предлагает их простой интеграции в базовый процесс архивации.

Также статья «Архиваторы под рентгеном» заложила основу для понимания того, что надёжность архива в большей степени определяется корректностью реализации современного криптоалгоритма и стойкостью пароля [1].

На основании проведённого анализа литературы сформулированы следующие гипотезы:

H1: Комбинация встроенных средств ОС Windows позволяет создать решение для запароливания архивов, криптостойкость которого может быть сопоставима с реализациями в современных сторонних архиваторах.

H2: Стойкость полученного архива к атаке полным перебором имеет экспоненциальную зависимость от длины пароля и размера криптографической соли.

Методы исследования

Исследование является экспериментально-прикладным. Его основная задача не только теоретический анализ возможностей встроенных средств Windows, но и практическая разработка, тестирование и количественная оценка работоспособности предложенной методики в контролируемых условиях.

Характеристика выборки: программная платформа – Windows 11 со всеми актуальными обновлениями с ключевыми компонентами, такими как среда PowerShell и платформа .NET, предоставляющая доступ к пространству имён System.Security.Cryptography. Тестовые данные – синтетический тестовый файл фиксированного размера (10 МБ).

Методы сбора данных: прямое измерение, включающее в себя фиксацию времени выполнения ключевых этапов процесса, моделирование атаки с помощью самодельного скрипта, сравнительный анализ конфигураций для сопоставления результатов времени взлома для различных комбинаций параметров. При этом следует учитывать, что полученные оценки времени перебора относятся исключительно к реализации атаки средствами PowerShell на одном CPU. Использование современных GPU и специализированного ПО может сократить время взлома на несколько порядков, что необходимо учитывать при интерпретации абсолютных значений.

Описание процедуры проведения исследования:

Для генерации тестовых данных в директории D:\TestArchive был создан набор текстовых файлов минимального объёма.

Далее с помощью скриптов было реализовано два параллельных тестирования, отличающихся ключевым криптографическим параметром – количеством итераций алгоритма выработки ключа PBKDF2.

Скрипт создавал ZIP-архив из тестовых данных с помощью Compress-Archive [5]. Полученный архив шифровался алгоритмом AES-256, а ключ шифрования вырабатывался из пароля ("9"*length) с использованием PBKDF2 с количеством итераций 100 или 10000 [6]. Данное количество итераций выбрано исключительно для демонстрации разницы в производительности и не являются стандартом безопасности на 2025/2026 год. Согласно актуальным рекомендациям OWASP (Password Storage Cheat Sheet), для алгоритма PBKDF2-HMAC-SHA256 минимальное рекомендуемое количество итераций составляет 310 000 (а в последних обновлениях руководства рекомендуется не менее 600 000) для обеспечения адекватной защиты от атак с использованием GPU [8]. Далее запускался алгоритм, систематически генерирующий все возможные комбинации цифр заданной длины. Для каждой сгенерированной комбинации из заголовка зашифрованного файла извлекалась криптографическая соль и вектор инициализации, выполнялась попытка выработки ключа и последующего расшифрования, а корректность пароля проверялась по успешности дешифрования и наличию валидной сигнатуры файла. Фиксировалось общее время от начала перебора до успешного подбора пароля, подсчитывалось общее количество попыток, а также средняя скорость перебора, как отношение количества попыток ко времени.

Методы обработки данных: сравнительный и качественный анализ, а также экстраполяция, где на основе полученных данных и известной формулы сложности перебора будет построена оценочная таблица, демонстрирующая теоретическое время взлома для более широкого алфавита.

Ход исследования

Для подготовки рабочего окружения перед выполнением основных операций скрипта если не существует папка по нужному пути, то создадим её:

```
$testDataPath = "D:\TestArchive"
if (-not (Test-Path $testDataPath)) {
    New-Item -ItemType Directory -Path $testDataPath -Force
}
```

```
$alphabetDigits = "0123456789"
```

Функцией Create-SecureArchive создадим защищённый архив в два этапа. Сначала стандартное сжатие средствами Compress-Archive, затем шифрование полученного файла алгоритмом AES-256, функции для 10000:

```
function Create-SecureArchive {
    param(
        [string]$SourceFolder,
        [string]$OutputPath,
        [string]$Password
    )

    $tempZip = Join-Path $env:TEMP "temp_$(Get-Random).zip"

    try {
        Compress-Archive -Path "$SourceFolder\*" -DestinationPath $tempZip -Force

        $bytes = [System.IO.File]::ReadAllBytes($tempZip)
        $salt = New-Object byte[] 32
        (New-Object
            System.Security.Cryptography.RNGCryptoServiceProvider).GetBytes($salt)

        $key = [System.Security.Cryptography.Rfc2898DeriveBytes]::new(
            $Password, $salt, 10000,
            [System.Security.Cryptography.HashAlgorithmName]::SHA256
        ).GetBytes(32)

        $aes = [System.Security.Cryptography.Aes]::Create()
```

Рубрика 2. Методы и системы защиты информации, информационная безопасность

```
$aes.Key = $key
$aes.GenerateIV()

$ms = New-Object System.IO.MemoryStream
$cs = New-Object System.Security.Cryptography.CryptoStream(
    $ms, $aes.CreateEncryptor(),
[System.Security.Cryptography.CryptoStreamMode]::Write)

$ms.Write($salt, 0, $salt.Length)
$ms.Write($aes.IV, 0, $aes.IV.Length)
$cs.Write($bytes, 0, $bytes.Length)
$cs.Close()

[System.IO.File]::WriteAllBytes($OutputPath, $ms.ToArray())

} finally {
    if (Test-Path $tempZip) { Remove-Item $tempZip -Force }
}
}
```

Test-Password для проверки корректности пароля путём попытки расшифровки и верификации целостности архива:

```
function Test-Password {
    param(
        [string]$EncryptedFile,
        [string]$Password
    )

    try {
        $bytes = [System.IO.File]::ReadAllBytes($EncryptedFile)
        $salt = $bytes[0..31]
        $iv = $bytes[32..47]
        $encryptedData = $bytes[48..($bytes.Length - 1)]

        $key = [System.Security.Cryptography.Rfc2898DeriveBytes]::new(
            $Password, $salt, 10000,
[System.Security.Cryptography.HashAlgorithmName]::SHA256
        ).GetBytes(32)

        $aes = [System.Security.Cryptography.Aes]::Create()
        $aes.Key = $key
        $aes.IV = $iv

        $ms = New-Object System.IO.MemoryStream
        $cs = New-Object System.Security.Cryptography.CryptoStream(
            $ms, $aes.CreateDecryptor(),
[System.Security.Cryptography.CryptoStreamMode]::Write)

        $cs.Write($encryptedData, 0, $encryptedData.Length)
        $cs.Close()

        $decryptedBytes = $ms.ToArray()
        $tempFile = Join-Path $env:TEMP "test_$(Get-Random).zip"
        [System.IO.File]::WriteAllBytes($tempFile, $decryptedBytes)

        try {
            $archive = [System.IO.Compression.ZipFile]::OpenRead($tempFile)
            $archive.Dispose()
            Remove-Item $tempFile -Force
            return $true
        } catch {
            Remove-Item $tempFile -Force -ErrorAction SilentlyContinue
            return $false
        }
    }
}
```

```

    }
} catch {
    return $false
}
}

```

Crack-PasswordBruteForce необходима для моделирования атаки методом полного перебора с использованием Test-Password для 10000 итераций:

```

function Crack-PasswordBruteForce-Unlimited {
    param(
        [string]$EncryptedFile,
        [string]$Chars = "0123456789",
        [int]$MaxLength = 4
    )

    $startTime = Get-Date
    $attempts = 0
    $foundPassword = $null

    for ($len = 1; $len -le $MaxLength; $len++) {
        $indices = New-Object int[] $len
        do {
            $password = ""
            for ($i = 0; $i -lt $len; $i++) {
                $password += $Chars[$indices[$i]]
            }

            $attempts++

            if (Test-Password -EncryptedFile $EncryptedFile -Password $password)
            {
                $foundPassword = $password
                break
            }

            $carry = 1
            for ($i = $len - 1; $i -ge 0 -and $carry -gt 0; $i--) {
                $indices[$i] += $carry
                if ($indices[$i] -ge $Chars.Length) {
                    $indices[$i] = 0
                    $carry = 1
                } else {
                    $carry = 0
                }
            }

        } while ($carry -eq 1 -and -not $foundPassword)

        if ($foundPassword) { break }
    }

    $time = (Get-Date) - $startTime
    @{
        Password = $foundPassword
        Attempts = $attempts
        Time = $time
        Speed = if ($time.TotalSeconds -gt 0) { [math]::Round($attempts /
$хime.TotalSeconds, 2) } else { 0 }
    }
}

```

Для проверки меньшего количества солей в функции Create-SecureArchive заменим 10000 на 100. Также заменим функции Test-Password и Crack-PasswordBruteForce на Crack-

Рубрика 2. Методы и системы защиты информации, информационная безопасность

Password-HighSpeed, которая является функцией для высокоскоростного перебора паролей с оптимизированной проверкой с помощью сигнатуры для 100 итераций:

```
function Crack-Password-HighSpeed {
    param(
        [string]$EncryptedFile,
        [string]$Chars = "0123456789",
        [int]$MaxLength = 4
    )

    $bytes = [System.IO.File]::ReadAllBytes($EncryptedFile)
    $salt = $bytes[0..31]
    $iv = $bytes[32..47]
    $encryptedData = $bytes[48..($bytes.Length - 1)]

    $startTime = Get-Date
    $attempts = 0
    $foundPassword = $null

    for ($len = 1; $len -le $MaxLength; $len++) {
        $indices = New-Object int[] $len
        do {
            $password = ""
            for ($i = 0; $i -lt $len; $i++) {
                $password += $Chars[$indices[$i]]
            }

            $attempts++

            try {
                $key = [System.Security.Cryptography.Rfc2898DeriveBytes]::new(
                    $password, $salt, 100,
                    [System.Security.Cryptography.HashAlgorithmName]::SHA256
                ).GetBytes(32)

                $aes = [System.Security.Cryptography.Aes]::Create()
                $aes.Key = $key
                $aes.IV = $iv

                $ms = New-Object System.IO.MemoryStream
                $cs = New-Object System.Security.Cryptography.CryptoStream(
                    $ms, $aes.CreateDecryptor(),
                    [System.Security.Cryptography.CryptoStreamMode]::Write)

                $cs.Write($encryptedData, 0, $encryptedData.Length)
                $cs.FlushFinalBlock()

                $decryptedBytes = $ms.ToArray()

                if ($decryptedBytes.Length -ge 4 -and
                    $decryptedBytes[0] -eq 0x50 -and
                    $decryptedBytes[1] -eq 0x4B -and
                    $decryptedBytes[2] -eq 0x03 -and
                    $decryptedBytes[3] -eq 0x04) {
                    $foundPassword = $password
                    break
                }
            } catch {
                # Ошибка дешифрования - неверный пароль
            }
        } while ($true)

        $carry = 1
    }
}
```

```

        for ($i = $len - 1; $i -ge 0 -and $carry -gt 0; $i--) {
            $indices[$i] += $carry
            if ($indices[$i] -ge $Chars.Length) {
                $indices[$i] = 0
                $carry = 1
            } else {
                $carry = 0
            }
        }

    } while ($carry -eq 0 -and -not $foundPassword)

    if ($foundPassword) { break }
}

$time = (Get-Date) - $startTime
$speed = if ($time.TotalSeconds -gt 0) { [math]::Round($attempts /
$time.TotalSeconds, 2) } else { 0 }

@{
    Password = $foundPassword
    Attempts = $attempts
    Time = $time
    Speed = $speed
    TotalCombinations = [math]::Pow($Chars.Length, $MaxLength)
}
}

```

Результаты исследования

В ходе исследования сравнивались две конфигурации шифрования, отличающиеся количеством итераций алгоритма выработки ключа PBKDF2, что напрямую влияет на скорость атаки методом полного перебора. Отметим, что полученное время справедливо только для атаки «скрипт против скрипта».

Длина	Алфавит	Комбинации	ВремяВзлома_сек	Попыток	Скорость	НайденныйПароль
1	Цифры (10)	10	0,01	10	909.12 попыток/сек	9
2	Цифры (10)	100	0,07	110	1511.14 попыток/сек	99
3	Цифры (10)	1000	0,55	1110	2017.82 попыток/сек	999
4	Цифры (10)	10000	5,29	11110	2098.57 попыток/сек	9999

Рис. 1. Результаты алгоритма на 100 итераций (маленькая соль)

Длина	Алфавит	Комбинации	ВремяВзлома_сек	Попыток	Скорость	НайденныйПароль
1	Цифры (10)	10	0,23	10	43.02 попыток/сек	9
2	Цифры (10)	100	2,41	110	45.69 попыток/сек	99
3	Цифры (10)	1000	23,79	1110	46.65 попыток/сек	999
4	Цифры (10)	10000	249,27	11110	44.57 попыток/сек	9999

Рис. 2. Результаты алгоритма на 10000 итераций (большая соль)

Таблица 1 – Сравнительная таблица результатов эксперимента

Длина пароля	Алфавит	Количество комбинаций	Время взлома 100 итераций PBKDF2, с	Время взлома 10000 итераций PBKDF2, с
1	цифры (10)	10	0,01	0,23
2	цифры (10)	100	0,07	2,41
3	цифры (10)	1000	0,55	23,79
4	цифры (10)	10000	5,29	249,27

Важно отметить, что полученные оценки времени перебора относятся исключительно к реализации атаки средствами PowerShell на одном центральном процессоре (CPU), и при использовании современного специализированного аппаратного обеспечения (например,

Рубрика 2. Методы и системы защиты информации, информационная безопасность

графических процессоров уровня NVIDIA RTX 4090) и оптимизированного программного обеспечения для взлома (такого как Hashcat), скорость перебора хэшей PBKDF2-НМАС-SHA256 может достигать десятков миллионов попыток в секунду. В таком сценарии время взлома 4-значного числового пароля сократится с сотен секунд до долей секунды, а для 8-значного пароля составит от нескольких секунд до минут. Это подчеркивает критическую необходимость использования длинных (от 12 символов) и сложных паролей, а также применения максимального числа итераций, рекомендованного современными стандартами.

Заключение

В работе было проведено экспериментально-прикладное исследование, направленное на решение актуальной практической задачи: преодоление функционального ограничения встроенных средств архивации ОС Windows, а именно отсутствия поддержки парольной защиты в утилите Compress-Archive. В качестве методологического подхода была предложена и реализована гибридная методика, комбинирующая стандартное средство сжатия с криптографическими возможностями платформы .NET Framework, доступными через PowerShell.

В ходе исследования был разработан, реализован и протестирован скриптовый комплекс, включающий функции создания запароленных архивов на основе алгоритма AES-256, а также инструменты для моделирования атаки методом полного перебора. Эксперимент проводился в контролируемых условиях с варьированием ключевых параметров безопасности: длины пароля и вычислительной сложности алгоритма выработки ключа PBKDF2 (100 и 10000 итераций). Собранные количественные данные позволили построить модели и провести экстраполяцию для оценки стойкости защиты в различных сценариях.

Результат проверки гипотез:

1. Гипотеза Н1 подтвердилась. Комбинация встроенных инструментов Windows (PowerShell Compress-Archive и криптографических библиотек .NET System.Security.Cryptography) действительно позволяет создать работоспособное и криптостойкое решение для защиты архивов. Стойкость системы при использовании алгоритма AES-256 с демонстрационными параметрами (длинная соль, 10 000 итераций PBKDF2) сопоставима с реализациями в современных сторонних архиваторах, основанных на тех же криптографических примитивах.

2. Гипотеза Н2 получила убедительное количественное подтверждение. Экспериментально установлена чёткая экспоненциальная зависимость времени взлома от длины пароля. Каждый дополнительный символ в пароле увеличивает сложность перебора в степени, равной мощности алфавита (в 10 раз для цифр, в 62 раза для алфавита букв и цифр) [7]. Кроме того, критическое влияние параметров криптографии продемонстрировано на примере количества итераций PBKDF2: их снижение с 10 000 до 100 привело к увеличению скорости перебора в 44,5 раза, что подчёркивает важность корректной настройки всех элементов системы безопасности, а не только выбора сложного пароля.

Данная работа открывает перспективные направления для дальнейших исследований. К примеру, можно продолжить развивать вопрос безопасности встроенных средств архивирования ОС Windows, рассмотрев его с использованием других методов. Также есть возможность для исследования этой темы для усовершенствования уже предложенного в работе эксперимента, использования различных режимов шифрования или разработка графического интерфейса для более комфортного повседневного использования.

Результаты данной работы подтверждают, что встроенные средства ОС представляют собой мощный и недостаточно используемый ресурс для построения гибких, безопасных и не зависящих от стороннего ПО решений в области защиты данных.

Библиографический список

1. Архиваторы под рентгеном. – Текст : электронный // Хакер : журнал. – 2006. – 12 дек. – URL: <https://xakep.ru/2006/12/12/35719/> (дата обращения: 01.11.2025).
2. Защищенное хранение данных и полнодисковое шифрование / Е. К. Алексеев, Л. Р. Ахметзянова, А. А. Бабуева, С. В. Смышляев. – Текст : электронный // Прикладная

- дискретная математика. – 2020. – № 4 (50). – С. 95–110. – URL: <https://cyberleninka.ru/article/n/zaschisyonnoe-hranenie-dannyh-i-polnodiskovoe-shifrovanie> (дата обращения: 02.11.2025).
3. How to password protect a zip file in Windows 11? – Текст : электронный // Microsoft Community Hub : сайт. – URL: <https://techcommunity.microsoft.com/discussions/windows11/how-to-password-protect-a-zip-file-in-windows-11/4408591> (дата обращения: 01.11.2025).
 4. How to Password Protect a Zip File on Windows 11/10 PC. – Текст : электронный // SYSGEEKER : сайт. – URL: <https://www.sysgeeker.com/blog/password-protect-a-zip-file-on-windows-pc.html> (дата обращения: 01.11.2025).
 5. Compress-Archive (Microsoft.PowerShell.Archive). – Текст : электронный // Microsoft Learn : официальная документация. – URL: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.archive/compress-archive?view=powershell-7.5> (дата обращения: 01.11.2025).
 6. Симметричный алгоритм блочного шифрования Advanced Encryption Standard. – Текст : электронный // Хабр : портал. – URL: <https://habr.com/ru/articles/534620/> (дата обращения: 03.11.2025).
 7. Сколько времени требуется для взлома пароля в 2024 году? – Текст : электронный // Comss : портал о кибербезопасности. – URL: <https://www.comss.ru/page.php?id=13587> (дата обращения: 05.11.2025).
 8. OWASP Password Storage Cheat Sheet. – Текст : электронный // OWASP Foundation : официальная документация. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата обращения: 10.11.2025).

References

1. Archivers Under the X-Ray // Hacker. URL: <https://xakep.ru/2006/12/12/35719/> (accessed: 01.11.2025).
2. Alekseev, E. K., Akhmetzyanova, L. R., Babueva, A. A., & Smyshlyaev, S. V. (2020). *Secure Data Storage and Full-Disk Encryption*. PDM, (49). URL: <https://cyberleninka.ru/article/n/zaschisyonnoe-hranenie-dannyh-i-polnodiskovoe-shifrovanie> (accessed: 02.11.2025).
3. How to password protect a zip file in Windows 11? // Microsoft Community Hub. URL: <https://techcommunity.microsoft.com/discussions/windows11/how-to-password-protect-a-zip-file-in-windows-11/4408591> (accessed: 01.11.2025).
4. How to Password Protect a Zip File on Windows 11/10 PC // SYSGEEKER. URL: <https://www.sysgeeker.com/blog/password-protect-a-zip-file-on-windows-pc.html> (accessed: 01.11.2025).
5. Compress-Archive (Microsoft.PowerShell.Archive) - PowerShell | Microsoft Learn // Microsoft Learn. URL: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.archive/compress-archive?view=powershell-7.5> (accessed: 01.11.2025).
6. Symmetric Block Encryption Algorithm Advanced Encryption Standard // Habr. URL: <https://habr.com/ru/articles/534620/> (accessed: 03.11.2025).
7. How Long Does It Take to Crack a Password in 2024? // Comss. URL: <https://www.comss.ru/page.php?id=13587> (accessed: 05.11.2025).
8. OWASP Password Storage Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (accessed: 05.11.2025).

Рубрика 2. Методы и системы защиты информации, информационная безопасность

Информация об авторах

Бережная Полина Александровна – студентка 3 курса факультета комплексной безопасности топливно-энергетического комплекса ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, e-mail: palinaberezhnaya047@gmail.com

Мирчевски Кристина Далиборовна – студентка 3 курса факультета комплексной безопасности топливно-энергетического комплекса ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, e-mail: mirchevski23@gmail.com

BUILT-IN WINDOWS OS ARCHIVING TOOLS: SECURITY CONCERNS

Berezhnaya P. A.¹, Mirchevski K. D.¹

¹National University of Oil and Gas «Gubkin University», Moscow, Russian Federation

Abstract. *The article presents an experimental study of a limitation of the native Windows archiving tools: the lack of built-in password protection. The aim is to develop and test a method for creating cryptographically strong password-protected archives using only built-in OS components (PowerShell and .NET Framework), without third-party software. On a Windows 11 test bench, the standard Compress-Archive utility was combined with AES-256 symmetric encryption. The encryption key was derived with PBKDF2 using a configurable iteration count and a cryptographic salt. A custom PowerShell script was used to simulate a brute-force password attack. The results show that increasing password length substantially expands the search space: each additional character increases it by a factor of 10 for numeric passwords or 62 for alphanumeric passwords. Reducing the PBKDF2 iteration count from 10,000 to 100 increased the brute-force rate by a factor of 44.5, demonstrating the importance of correctly configuring all security parameters. The study also notes that modern specialized hardware can accelerate brute-force attacks by orders of magnitude, so long passwords and current recommended iteration counts are required in practice. The hypotheses were confirmed. The proposed approach can provide cryptographic strength comparable in principle to third-party archivers when current cryptographic practices are followed. The method is practically relevant to system administrators, end users, and information-security education.*

Keywords: Windows, archiving, data security, PowerShell, encryption, AES, built-in tools.

Information about the authors

Berezhnaya Polina Aleksandrovna – 3rd year student, Faculty of Integrated Safety of the Fuel and Energy Complex, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, e-mail: palinaberezhnaya047@gmail.com

Mirchevski Kristina Daliborovna – 3rd year student, Faculty of Integrated Safety of the Fuel and Energy Complex, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, e-mail: mirchevski23@gmail.com