Stealing From Thieves: Breaking IonCube VM to RE Exploit Kits

Mohamed Saher (@halsten)
About @halsten

• Reverse Engineering
• Automation of RE tasks
• Virtualization
• Regular project-euler problem solver (ranked #1 locally)
• Old crackmes writer and solver
Contents

• What is ionCube?
• Why Protect?
• How does it work?
• VM Architecture
• VM Internals
• ionCube Loader (SAMPLE)
• Breaking ionCube
  – Extracting RAW DATA
  – Validating RAW DATA
  – Processing RAW DATA
  – Interpreting the Header
  – Interpreting the Extra Header
• Conclusion
• Q & A
Not Covered

• Recovering the license file
• Cracking the license decryption algorithm
  – DRM law
• Decompilation of VM Handlers and restoring original PHP source
  – Out of scope
What is ionCube?

• Packer/Compressor
What is ionCube?

- Packer/Compressor
- Protector/Virtualizer
Why Protect?

- Intellectual property
Why Protect?

- Intellectual property
  - Algorithm implementation
Why Protect?

• Intellectual property
  – Algorithm implementation
  – Serial checking routines
Why Protect?

• Intellectual property
  – Algorithm implementation
  – Serial checking routines
  – Hard-coded configurations
Why Protect?

• Intellectual property
  – Algorithm implementation
  – Serial checking routines
  – Hard-coded configurations
  ...

Why Protect?

• Intellectual property
  – Algorithm implementation
  – Serial checking routines
  – Hard-coded configurations
  ...

• Public distribution without modification to the original source
PE Packer vs. PHP Encoder

• Traditional PE Packers compress/protect the x86 code and uses its stub to decompress/unprotect it back in during execution

• PHP Encoders has to rely on the ZEND technology (php->zend_opcodes)
How does it work? (Compilation)

Protection

- Transform to Intermediate Language (IL)
- Assembled to Bytecode
- Obfuscate Bytecode

Protected PHP File 1
Protected PHP File 2
Protected PHP File N
How does it work? (Run-Time)
VM Architecture

• Stack based VM (example: .NET, Java)
• Byte Code is obfuscated after compilation
• Uses some crypto for VM header and parameter encryption
• Uses Zend Engine
VM Internals

- Crypto used within the encoder and the VM
  - Custom Base64
  - Adler32
  - CRC32
  - SHA-1
  - MD5
  - BlowFish (Counter Mode Encryption [CTR])
  - Modified Mersenne Twister
Example of a Protected PHP File

```php
<?php

if(!extension_loaded('ionCube Loader')){
    $__oc=strtolower(substr(PHP_UNAME(),0,3));
    $__ln='/ioncube/ioncube_loader_'.substr($__oc,'.').'.' . substr(PHP_VERSION(),0,3).
    (($__oc=='win')?'\dll':'.so');
    $__oid=$__id=realpath(ini_get('extension_dir'));
    $__here=dirname($__FILE__);if($__$oid[1]==':'){$__id=str_replace('\\', '/', substr($__id, 2));$__here=str_replace('\', '/', substr($__here, 2));}

    $__rd=str_repeat('..', substr($__id, 0, 1));
    $__i=strlen($__rd);
    while($__i--){if($__rd($__i)=='/'){$__lp=substr($__rd, 0, $__i).
        if(file_exists($__oid($__lp))){$__ln=$__lp;break;}}}
    @dl($__ln);else{echo('The file '.($__FILE__.' is corrupted.\n'');return 0;}}}if(function_exists('_il_exec')){return _il_exec();}
    echo('This encoded file cannot be run. Please run the file ioncube-loader-helper.php for more information.');return 0;

?>
```

4oV5E3tizCOGmZayKycyFfddNEYKcDQ2UctWQgi5wUMAYDSmMVeoLZpTJY1sb2ZS87vmUDNyJXy
u6mBqXBOY8uBDM8S9FpYpOU8H2UybP4eoySb3gsXR3LDVhZQOE547V1admAtDtg672Z0axEinz
4Q0KK4ySJmQf/y74+9n0mQxv89e/3ORP/KEy9C7qQ57ANCP167ft8uwqnxmMG2B0FghtWsgbjWW
TRM9Hpx9RfSRUpbRfJyiWM77aOjZWR9XB2aJyxq/75a5+EXV17auQnQ2ZiQhbeeajCwKRwWP0X9
N8VmcedG2VriSa6TMSY++2C4zLx5FcRziK7DMb2vYBQA0IhN8SOiVv4t5lIzymywsmq9bHtAZLdU
62oKLWPotyYaB7R/+nSDX4s7Vwifp0nXJe8NQ5zI36p4UMmoZnHHKC/+oFab7U7rI4uC707fwrhr
b95eZu1QsG+TWFhNjn3Ao9UClGrvoye+Fil7xrg=
How does it work? (Internally)

• There’s only 1 way to find out, let’s see what the loader is doing under the hood
Breaking ionCube
Extracting the RAW DATA

• Decode the RAW DATA using a custom Base64 character set
  "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz+/"
and not
  "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

4+oV5E3tizCOGmZayKycyFdfdNEYKcDQ2UctWQgi5wUMAYDSmMVeOLZpTJY1sb2ZS87vmUDNyJXy
u6mBqXBOY8uBDM8S9FpfYpOU8H2YbP4eoySb3gsXR3LRDVhZQOE547VladmAtDtg6Z720axEinz
4Q0KK4ySJmQf/y74+9n0mQxv89e/3ORP/KEy9C7qQ57ANCp167ft8uwqnxmMG2B0Fgh-twVs5gbjWW
TRM9HpX9RfSRUpbRfJyiWM77aOjzW9XB2eAJyxdT/T5a5+EXV17auGnO2ZiQhbeeaJcWKwW'F0X9
N8VmcEdG2VrIsa6TMSY++2C4zLx5FcRziK7DMb2vYBQA0InR8S0IvV4t5JIZumywsmq9bHtAZLdU
62oKLPotyYaB7R/+nSDX4s7Vwifp0nXJe8NQ5zI36p4UmmOZnHHKC/+oFab7U7rI4uC707fwrhr
b95e2u1QsG+TWFhNjn3Ao9UC1Grvoye+fIL7xreq=
Breaking ionCube
Extracting the RAW DATA

• Check for encoded VM restrictions and rules
• Header size (<?php //0) -> 10 bytes + 4 bytes (size of loader)
• Determine the starting offset of the loader
Breaking ionCube
Extracting the RAW DATA

• Get PHP version (DWORD)
  – Compare with HARD-CODED values (BINARY MODE)
    • 0xDEADC0DE
    • 0x3FBC2883
    • 0x217582F
    • 0x149FEC13
    • 0x67A6BF45
    • 0x9EB67AC2
Breaking ionCube
Extracting the RAW DATA

- Compare against other HARDCODED values (BASE64 MODE)
  - 0y4h
  - BrWN
  - 4+oV
  - HR+c
  - mdgs
Breaking ionCube
Validating the RAW DATA

- Read a DWORD for the VM version (0x00) XOR the value with 0x2853CEF2 and compare with HARDCODED values
  - \( \text{dwVer} = \text{ReadDWORD()} \oplus 0x2853CEF2 \)
    - 0x17EFE61 (v1)
    - 0x2A4496DD (v2)
    - 0x3CCC22E1 (v3)
    - 0x4FF571B7 (v4)
    - 0xA0780FF1 (v5)
    - 0xB6E5B430 (v6)
    - 0xF6FE0E2C (v7)
Breaking ionCube
Process the RAW DATA

• Calculate dwFileSizeKey (DWORD)
  – \(\text{dwFileSizeKey} = ((\text{dwRawBinaryDataSize} + 12321)^{0x23958CDE})\)

• Read Header Information (struct)
  – dwHeaderFileSizeKey (DWORD +0x00)
  – dwHeaderSize (DWORD +0x04)
  – dwHeaderKey (DWORD +0x08)
Breaking ionCube
Process the RAW DATA

• Calculate Header Size using the following formula
  – dwCalculatedHeaderSize = (((dwHeaderSize ^ 0x184FF593) + (-0x0C21672E)) ^ dwHeaderKey)
  – dwFillData1 (DWORD +0x0C)
  – dwFillData2 (DWORD +0x10)
  – dwFillData3 (DWORD +0x14)

• dwFillData1/dwFillData2/dwFillData3 (encoded during runtime with 0xFF “<“)

• Calculate Header File Size Key
  – dwCalculatedHeaderFileSizeKey = (dwHeaderFileSizeKey ^ dwHeaderKey)
• Validate Key
  – If (dwFileSizeKey != dwCalculatedHeaderFileSizeKey)
    • Difference -> ABS(dwFileSizeKey - dwCalculatedHeaderFileSizeKey)
    • Recover the Key
      – dwNewCalculatedHeaderFileSizeKey = ((dwCalculatedHeaderFileSizeKey – 12321) ^ 0x23958CDE)
• Initialize MT PRNG with dwHeaderKey
Breaking ionCube
Process the RAW DATA

• Read the Header Data and Checksum values.
• Header consists of multiple chunks (struct)
  – Parse Header Chunks
    • Loop while (dwCounter <= dwCalculateHeaderSize)
      – dwChunkFlag (BYTE)
      – dwChunkSize (BYTE)
  – Read the MD5 checksum of the Raw Data (0x10 BYTES)
Breaking ionCube
Process the RAW DATA

• Validate ADLER32 checksum for the encoded VM
  — START: EncodedVM + 0x04
  — END: EncodedVM.EOS

• Extract CRC from Header
  — dwCRC == dwCalculatedADLER32
Breaking ionCube
Process the RAW DATA

• Decrypt Chunk Key using the following algorithm
  foreach (BYTE dwB in dwMD5Checksum) {
    ROR(dwB, 3)
  }

• Decrypt Header with the following algorithm
  while (Header.POSITION != EOS) {
    while (dwMD5Checksum.POS != EOS) {
      x = ReadDWORD()
      y = dwMD5Checksum.ReadBYTE()
      z = (x ^ Rand_MT(0xFF) ^ y)
    }
  }

• At this point we have extracted ionCube Header in Binary format
Breaking ionCube
Interpreting the Header

• Read dwVersionData for the Header version (DWORD)
• Read dwMinimumLoaderVersion (DWORD)
• Read dwObfuscationFlags
  – Decode dwObfuscationFlags
    • VARIABLES -> 0x0004
    • FUNCTIONS -> 0x0008
• Read dwHeaderCustomLoaderEventMessagesCount (DWORD)
• Read a fixed sized string for szObfuscationHashSeed with fixed size of
dwHeaderCustomLoaderEventMessagesCount
Breaking ionCube
Interpreting the Header

• Try to extract dwByteCodeKey
  – Doesn’t exist?
    • Assume a HARD-CODED value of 0x363432
  – Exists?
    • Read it normally
      – If (dwByteCodeKey == 0x92A764C5)
        » SPECIAL CASE: LicenseFile(+EnforceLicense)
          • License File exists?
            • YES: GOOD
            • NO: Could be calculated and recovered
Breaking ionCube
Interpreting the Header

• Read dwIncludedXORKey (HARD-CODED value: 0xE9FC23B1)
• Read dwNumberOfStructsToRead which will specify how many structures to read based on the encoding of the original file
  – LicenseString (restricted by the value of dwSize)
    • dwDummy (DWORD)
    • dwSize (DWORD)
  – Check for DisableCheckingofLicenseRestriction (pointed by dwDummy3)
    • dwDummy1 (DWORD)
    • dwDummy2 (DWORD)
    • dwDummy3 (DWORD)
Breaking ionCube
Interpreting the Header

• Check for LicensePassphrase (restricted by the value of dwSize)
  – dwDummy (DWORD)
  – dwSize (DWORD)

• Check if there is a CustomErrorCallback file (restricted by the value of dwSize)
  – dwDummy (DWORD)
  – dwSize (DWORD)

• Check if there is a CustomErrorCallbackHandler (restricted by the value of dwSize)
  – dwDummy (DWORD)
  – dwSize (DWORD)
Breaking ionCube
Interpreting the Header

• Check for EnableAutoPrependAppendFile (pointed by dwDummy3)
  – dwDummy1
  – dwDummy2
  – dwDummy3

• Skip 2 dummy DWORD and a calculated number of bytes
  – dwCalculatesBytes = \text{ABS}(\text{dwNumberOfStructsToRead} - 5)
Breaking ionCube
Interpreting the Header

* Decode the CustomErrorMessage (the following) with the later algorithm
  - Corrupt-file
  - Expire-file
  - No-permissions
  - Clock-skew
  - Untrusted-extension
  - License-not-found
  - License-corrupt
  - License-expired
  - License-property-invalid
  - License-server-invalid
  - Unauth-including-file
  - Unauth-included-file
  - Unauth-append-prepend-file
Breaking ionCube
Interpreting the Header

• Read dwNumberOfCustomizedErrorMessages which will determine how many structs to read later

• Loop through dwNumberOfCustomizedErrorMessages and read the struct
  – dwCustomErrorMsgID (BYTE)
  – szCustomErrorMsg
    • WARNING: NULL-TERMINATED strings (skip ‘\’0’

Breaking ionCube
Interpreting the Header

• Decode IncludeFileRestrictions
  – Read dbNumberOfIncludeRestrictionsEntriesToRead (BYTE)
  – Loop through dbNumberOfIncludeRestrictionsEntriesToRead and read 2 sets of arrays of structs
    • Read dbDummy (BYTE) [NOT IMPORTANT]
    • Set 1 (IncludeKey)
    • Set 2 (IncludeKeyHandler)
Breaking ionCube
Interpreting the Header

• Both Set 1 and 2 need to be decoded using the following algorithm
  – Read wSize (WORD)
  – Calculate $Z = (wSize \ ^\ dwIncludeXORKey) \ & \ 65535$
  – Using the calculated $Z$ we can extract the full data and fully decode it using the following algorithm
    Do {
      a = ReadDWORD()
      b = (a \ ^\ dwIncludedKey)
    } while (!EOS)
Breaking ionCube
Interpreting the Header

• Read `dbNumberOfServerRestrictedItems` (BYTE)
• Loop through `dbNumberOfServerRestrictedItems` and read a struct
  – Read `dbNumberOfRows` (BYTE)
  – Read `dbNumberOfColumns` (BYTE)
    • Loop through `dbNumberOfColumns`
      – Read `dbDataType` (BYTE)
        » Decode `dbDataType`
        • 0 -> IP
        • 1 -> MAC
        • 3 -> NOT IMPORTANT
        • 4 -> DOMAIN
Breaking ionCube
Interpreting the Header

• IP
  – Read dbNumberOfIPEntries
    • Loop through dbNumberOfIPEntries
      – Read dbUseNetMask
        » 0 -> will use netmask
        » 1 -> will not use netmask
      – Read IP Address in reverse order
        » dbIP4
        » dbIP3
        » dbIP2
        » dbIP1
      – Read netmask in reverse order
        » dbNetMask4
        » dbNetMask3
        » dbNetMask2
        » dbNetMask1
Breaking ionCube
Interpreting the Header

• MAC
  – Read dbNumberOfMACEntries
    • Loop through dbNumberOfMACEntries
      – Read szMAC (6 BYTES)
Breaking ionCube
Interpreting the Header

• Domain
  – Read dbNumberOfDomainEntries
    • Loop through dbNumberOfDomainEntries
      – Read szDomain (NULL-TERMINATED)
Breaking ionCube
Interpreting the Header

• Compute $dw_{\text{CalculatedAdler32}}$ for the encoded VM
  – Difference between extracted and calculated? (Un/modified)
Breaking ionCube
Interpreting the Extra Header

• Read 0x28 bytes which contains the Extra Header
• Read wMinorVersion (WORD)
• Read wMajorVersion (WORD)
Breaking ionCube
Interpreting the Extra Header

- Read dwPHPFlags
  - Decode dwPHPFlags
    - 0x001
    - 0x002
    - 0x004
    - 0x008
    - 0x010
    - 0x020 (allow run with untrusted extensions)
    - 0x040 (php5 body)
    - 0x080 (vm handlers are encrypted)
    - 0x100
    - 0x200 (obfuscate function names)
    - 0x400 (encrypt strings)
    - 0x800 (obfuscate strip line numbers)
    - 0x1000 (obfuscate variable names)
    - 0x2000 (encryption flag)
    - 0x4000
    - 0x8000
Breaking ionCube
Interpreting the Extra Header

• Read dbEncoderGenerationNumber (BYTE)
• Read dbEncoderMajorNumber (BYTE)
• Read dbEncoderMinorNumber (BYTE)
• Read dbEncoderEnhancementNumber (BYTE)
Breaking ionCube
Interpreting the Extra Header

• Read dwMemberID (DWORD) [registration data]
  
  If (license exists and license restrictions are enforced) {
    dwByteCodeKey = (0x363432 + RAND(_time()))
  }

Breaking ionCube
Interpreting the Extra Header

- Morph the dwByteCodeKey using the following algorithm
  
  If (dwServerRestrictionItems exists) {
    dwByteCode_MT_XORKey = (0x92492493 / 0x1000) * (dwByteCode_MT_InitKey / 0x1000)
    dwByteCode_MT_XORKey = ByteCode_MT_XORKey + ByteCode_MT_InitKey
    dwByteCode_MT_XORKey = (int)(dwByteCode_MT_XORKey / 4)
  }

  If (dwByteCode_MT_XORKey < 0) {
    dwByteCode_MT_XORKey++
  }

  dwByteCode_MT_XORKey = (dwByteCode_MT_XORKey - (13 * dwServerRestrictionItems))

  } Else {
    dwByteCode_MT_XORKey = dwByteCode_MT_InitKey
  }

  (((((dwByteCode_MT_InitKey * 0x92492493 >> 32) + dwByteCode_MT_InitKey) >> 2) – 13 * dwServerRestrictionItems)
Breaking ionCube
Interpreting the Extra Header

- Read dwCopyOfIncludedXORKey (DWORD)
- Read dwUnknown1 (DWORD)
- Read dwUnknown2 (DWORD)
- Read wUnknown3 (WORD)
- Read dbUnknown4 (BYTE)
- Read dbUnknown5 (BYTE)
- Read dwEvalTimeMinEncryption
  - dwEvalTimeMin = dwEvalTimeMinEncryption + 10233976199
- Read dwEvalTimeMaxEncryption
  - dwEvalTimeMax = dwEvalTimeMaxEncryption + 83941958

- 0x4AD70D0D -> 16.10.2009
- 0x740A9780 -> 11.09.2031
- CONST dwSecondsPerDay = 86400 (24h * 60m * 60s)
Conclusion

- VM uses a simplistic Stack based approach
- Encryption/Encoding methods are weak and easily broken
- Relies too much on XOR based encryption
- LOTS of HARD-CODED constants
- Loader is easily patched (no protection, x86 code easily read)
- Couple of bugs in the Loader through hand-crafted VM (exploitable?)
Q & A

• Mohamed Saher
• @halsten
• msaher@nsslabs.com