NyxEngine

SDK References

Contents

Introduction to NyxEngine

Steganography[1] is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. When it comes to digital steganography no stone should be left unturned in the search for viable hidden data. Although digital steganography is commonly used to hide data inside multimedia files, a similar approach can be used to hide data in archives as well. Steganography imposes the following data hiding rule: *Data must be hidden in such a fashion that the user has no clue about the hidden message or file's existence*. This can be achieved by either hiding existing packed content from all programs designed to unpack the selected file format, or adding new data to existing compressed files, so that the file's usability is unchanged. To discover this hidden information we must go into deep analysis of systems that have developed their own archive processors and see the implications of format specifications being interpreted differently across such solutions.

We have designed *NyxEngine* to ensure that no byte is left unchecked in the search for interesting archive data. Furthermore Nyx performs detailed data inspection by which it identifies possible vulnerabilities and corruptions in the binary content of archives. By integrating the *NyxEngine* as the top layer in archive processing, we can successfully detect and prevent all known and future vulnerability attack vectors against archive processors, thus effectively eliminating the possibility of archive bombs and other exploits. In addition to shielding against exploits, Nyx also searches for viable hidden data that was intentionally cloaked from sight using steganographic principles. And since the engine does detailed data inspection, it can correct vulnerabilities and recover files, making it a perfect archive preprocessor.

Nyx engine's exploit shield functionality checks the following archive areas: stored file name length and content, compression ratio, extract algorithm requirements, checksum tampering, multi-disk tampering, file entry duplication and other miscellaneous header data checks. Serving as a common denominator among all known archive processing solutions, Nyx classifies each instance of tampering in a functional group as vulnerabilities that affects that group.

By performing detailed checks and on-the-fly corrections, the maximum possible archive data is recovered and identified. This is the best way to find files that are present in the archive, but unreported in the archive header and to extract every possible bit from the archive. This method this works not only with unreported files, but with any kind of binary data present in the archive which isn't assigned to any of the file content.

The detailed file analysis provided by Nyx makes it possible to recover the maximum amount of damaged, corrupt and invalid data.

1

"Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means concealed writing."

Introduction to ZIP file format

The ZIP file format is one of the most common archive file formats used today. The format was originally created in 1986 by Phil Katz for PKZIP, and evolved from the previous ARC compression format by Thom Henderson. The PKZIP format is now supported by many software utilities other than PKZIP. Microsoft has included built-in ZIP support (under the name "compressed folders") in versions of its Windows operating system since 1998. Apple has included built-in ZIP support in Mac OS X 10.3 and later.

ZIP is a simple archive format that compresses every file separately. Compressing files separately allows individual files to be retrieved without reading through other data; in theory, it may allow better compression by using different algorithms for different files. A caveat to this is that archives containing a large number of small files end up significantly larger than if they were compressed as a single file, due to the fact that the data structures which store information on each individual file are stored uncompressed.

The ZIP file's contents comprise files and directories stored in arbitrary order. The files and directories are represented by file entries. The location of each file is indicated in a “central directory”, located at the end of the ZIP file.

Each file entry is introduced by a local header with information about the file such as the comment, file size and file name, followed by optional "extra" data fields, and then the possibly compressed, possibly encrypted file data. The "Extra" data fields are the key to the extensibility of the ZIP format. It is the "extra" fields that are exploited to support ZIP64 formats, WinZip-compatible AES encryption, and NTFS file timestamps. In theory there are many other extensions possible via this coded "extra" field.

The central directory consists of file headers holding, among other metadata, the file names and the relative offset in the archive of the local headers for each file entry. Each file entry is marked by a specific 4-byte "signature"; each entry in the central directory is likewise marked with a different particular 4-byte signature. ZIP file parsers typically look for the appropriate signatures when parsing a ZIP file. Due to the fact that the order of the file entries in the directory need not conform to the order of file entries in the archive, the format is non-sequential. There is no BOF or EOF marker in the ZIP spec. Instead, ZIP tools scan for the signatures of the various fields.

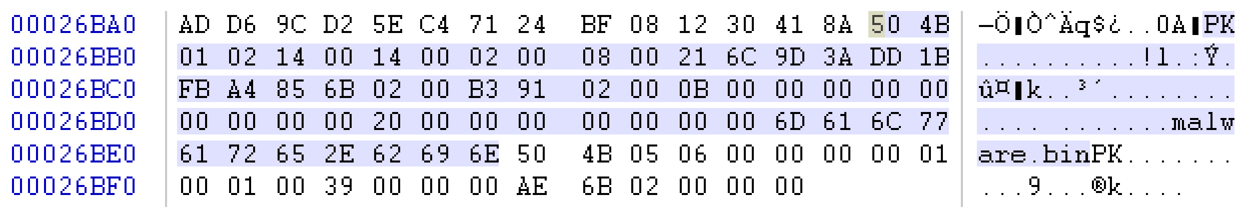
There are numerous ZIP tools available, and numerous ZIP libraries for various programming environments. Some of the libraries are commercial, some are not. Some are open source, some are not. WinZip is perhaps the most popular and famous ZIP tool - it runs primarily on Windows and is a user tool for creating or extracting ZIP files. WinRAR, IZarc, Info-zip, 7-zip are other tools, available on various platforms. Some of those tools have library or programmatic interfaces.

Introduction to steganography in ZIP archives

If we take a look at the ZIP central directory structure we can see the following:

```
typedef struct NYX_ZIP_CENTRALDIR{
      DWORD ZIPSignature;
      WORD ZIPVersion;
      WORD ZIPExtractVersion;
      WORD ZIPGeneralPurpose;
      WORD ZIPCompressionMethod;
      WORD ZIPLastFileModTime;
      WORD ZIPLastFileModDate;
      DWORD CRC32;
      DWORD FileCompressedSize;
      DWORD FileUncompressedSize;
      WORD FileNameStringLength;
      WORD ExtraDataLength;
      WORD FileCommentLength;
      WORD DiskNumberStart;
      WORD InternalFileAttributes;
      DWORD ExternalFileAttributes;
      DWORD RelativeOffsetOfLocalHeader;
      //Following this is data with variable size
}NYX_ZIP_CENTRALDIR, *PNYX_ZIP_CENTRALDIR;
```

First thing that comes to mind in order to achieve existing data steganography is setting the value of the field *FileNameStringLength* to zero. But that can't work because file name is written just after the central directory entry, and if we reset its length to zero the archive will become corrupt, since the next item of the central directory won't be located correctly. This is what one central directory entry looks like:

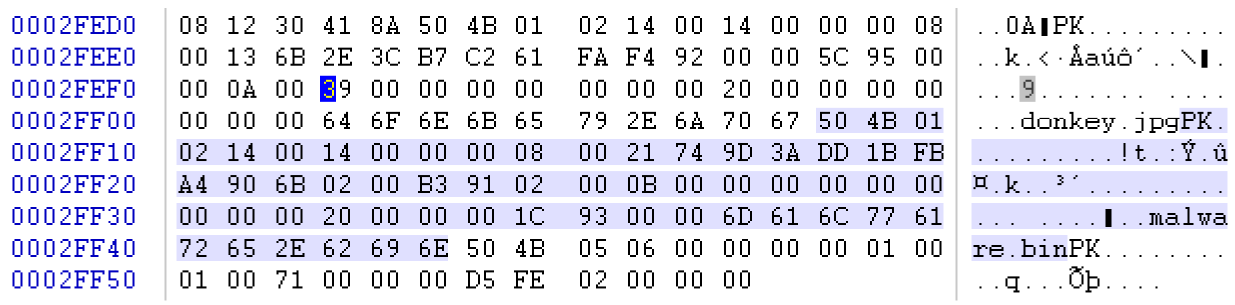


```
00026BA0  AD D6 9C D2 5E C4 71 24  BF 08 12 30 41 8A 50 4B  –Ö▮Ò^Äq$¿..0A▮PK
00026BB0  01 02 14 00 14 00 02 00  08 00 21 6C 9D 3A DD 1B  ..........!l.:Ý.
00026BC0  FB A4 85 6B 02 00 B3 91  02 00 0B 00 00 00 00 00  û¤▮k..³´........
00026BD0  00 00 00 00 20 00 00 00  00 00 00 00 6D 61 6C 77  .... .......malw
00026BE0  61 72 65 2E 62 69 6E 50  4B 05 06 00 00 00 00 01  are.binPK.......
00026BF0  00 01 00 39 00 00 00 AE  6B 02 00 00 00           ...9...®k....
```

Since the checksum check only ensures integrity of the compressed data, but not the header we can easily modify any member of the central directory entry structure. To make **malware.bin** vanish from the list of all programs that work with the ZIP file format we simply modify the file name. By changing the first character of the name (the byte 0x6D which is the letter 'm') to 0x00, we effectively hide this file from being listed by almost all programs that work with the ZIP file format. Only two programs are not fooled by this trick: WinZIP and 7Zip. Those two are the only programs that, regardless of the invalid name, show the file as an entry in their view. But how can changing the name work? Can it be reverted by some steganography program later? The answer is yes, because ZIP stores file names at two locations, once at the central directory entry and once at the local directory entry - and since only one of the two is modified, the other one can be used to revert the file to original state.

Even though this approach offers limited hiding capabilities, there are other more effective ways of hiding data in archives. One such way would be exploiting ZIP file format specifics to make data unreadable by archivers but leaving it present and its state easily reversible to the original. One such way would be utilization of extra data fields described in the PKWARE ZIP file format specification.

This field was introduced because of the need to store extra information about the file such as NTFS data streams, encryption information and other data utilized by applications that process this format. There are quite a few documented uses of this field in the standard itself. However due to the freedom offered by such data a structure, there are multiple ways to use it. For data hiding, you need only expand the extra field of one file to consume one or more of the files that follow it in the archive header. After this, correcting the fields ThisDiskItemEntries and DiskItemEntries ensures that the archive remains valid. Such archive modification would look like this:



```
0002FED0  08 12 30 41 8A 50 4B 01  02 14 00 14 00 00 00 08  ..0A▌PK.........
0002FEE0  00 13 6B 2E 3C B7 C2 61  FA F4 92 00 00 5C 95 00  ..k.<·Âaúô´..\▌.
0002FEF0  00 0A 00 39 00 00 00 00  00 00 00 20 00 00 00 00  ...9........ ....
0002FF00  00 00 00 64 6F 6E 6B 65  79 2E 6A 70 67 50 4B 01  ...donkey.jpgPK.
0002FF10  02 14 00 14 00 00 00 08  00 21 74 9D 3A DD 1B FB  .........!t.:Ý.û
0002FF20  A4 90 6B 02 00 B3 91 02  00 0B 00 00 00 00 00 00  ¤.k..³´.........
0002FF30  00 00 00 20 00 00 00 1C  93 00 00 6D 61 6C 77 61  ... ....▌..malwa
0002FF40  72 65 2E 62 69 6E 50 4B  05 06 00 00 00 00 01 00  re.binPK........
0002FF50  01 00 71 00 00 00 D5 FE  02 00 00 00              ..q...Õþ....
```

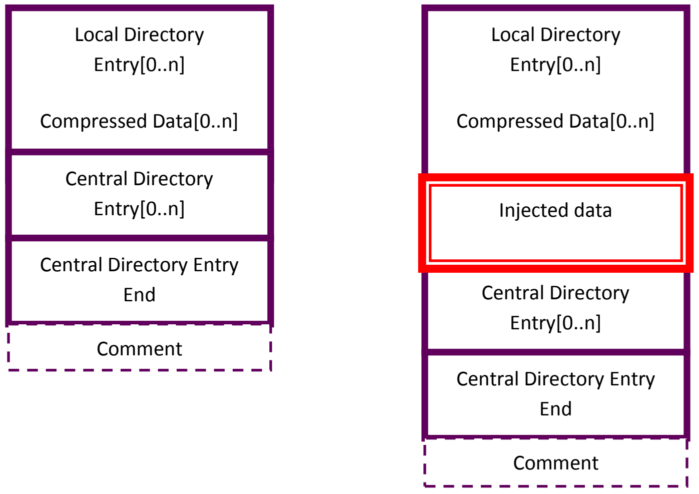
There are numerous ways one can hide files in a ZIP archive by modifying the archive headers. The next approach is even more elegant, as it involves a very small change to the header. But this time we are not talking about the central directory itself, but about the end of the central directory, which is the actual beginning of the archive, despite its position at the end of the file. Due to this positioning, you process a ZIP archive by searching for the signature pattern from the end of the archive. Here is the archive's signature structure:

```
typedef struct NYX_ZIP_CENTRALDIR_END{
	DWORD Signature;
	WORD DiskID;
	WORD ThisDiskID;
	WORD ThisDiskItemEntries;
	WORD DiskItemEntries;
	DWORD SizeOfCentralDir;
	DWORD LocationOfCentralDir;
	WORD ZipCommentLength;
}NYX_ZIP_CENTRALDIR_END, *PNYX_ZIP_CENTRALDIR_END;
```

To hide files, you simply change the LocationOfCentralDir pointer to point to the first file you want to be visible in the archive. After you do this, correcting the fields ThisDiskItemEntries and DiskItemEntries ensures that the archive remains valid. This procedure hides all files located prior to the file to which the modified LocationOfCentralDir points.

Reverting this is easy. One only needs to search backwards from LocationOfCentralDir for a central directory entry signature to find out how many files were hidden. And once those are found, correcting the fields ThisDiskItemEntries and DiskItemEntries restores the file to its original state.

The internal data structure of ZIP archives is similar to that the structure the file system uses to store data on drives. Here is the layout of files inside the archive:


Local Directory Entry[0..n]
Compressed Data[0..n]
Central Directory Entry[0..n]
Central Directory Entry End
Comment
Local Directory Entry[0..n]
Compressed Data[0..n]
Injected data
Central Directory Entry[0..n]
Central Directory Entry End
Comment

As we see from the figure above, files form an array of local directory entry structures followed by compressed data assigned to that local directory entry. Since the central directory entry end structure contains the pointer to the first central directory entry, it can be moved in any direction. By moving it up in a file we create space for data to be injected. One side effect of injecting data this way is that once new files are added to such an archive with an archiver application, injected data is automatically stripped during the unpacking process. This can be useful in some scenarios since it leads to protecting data from tampering.

One scenario for this kind of data protection would be packing random files inside a ZIP archive with a simple piece of malware that any antivirus detects. This way hidden data will be stripped by the antivirus itself, if the file is scanned for malware, which is useful if the file ends up in the wrong hands, because the hidden data will self destruct. A similar self-destruct approach can be used for OOXML file format, which is essentially a ZIP archive containing Office document data. The difference is that the Microsoft Office product line detects changes to the file and offers to automatically correct damage to the file, which if employed erases the hidden data.

This brief introduction to archive steganography shows that hiding data in archives is possible in numerous ways. Furthermore the hidden data quite often protects itself with unique self-destruct mechanisms, which is why we need to be careful when inspecting suspicious content if we want to truly find all viable data.

Steganography and file malformation security impacts

Multiple ways of introducing steganography to archive file formats impact computer system security. The effects of hiding files can go beyond steganography implications, creating a serious threat to archive parsing. Due to complexity of the process for creating archive software, and the inevitable ambiguity in the related documentation, it is not entirely uncommon for archive software to contain oversights that allow seemingly invalid files to be processed normally by some solutions. This is crucial for antivirus vendors to understand – since they want to support extraction of as many popular archive formats as possible. These slight differences in format specification make all the difference in overall security within a network, because if there is a single archive processing tool available that can decompress the archive within your network, its content must be fully inspected to make sure that that file is malware clean. Archive content can be scanned t the point of extraction, but your network will be much safer if scanning is done at the gateway so that the fewest possible malicious files reach their destination. This is particularly crucial for business organizations that only employ gateway scanners to protect their endpoints.

ReversingLabs Corporation has performed a series of in-depth checks to test the overall impact of archive file format malformations on security software. During our test, we intentionally malformed selected archive formats (ZIP, RAR, CAB, GZIP and 7ZIP) to the extent permitted by file format documentation, leaving each in such a state that packed content within it was considered valid and extractable by at least one popular format processing program. After malformation the files were subjected to multiple antivirus testing systems (available at www.virustotal.com). The results of our tests were used to create series of vulnerability security advisories available at the ReversingLabs website. Prior publishing them, ReversingLabs, in association with CERT-FI, contacted all affected security vendors, and helped those who responded to verify the implemented fixes. The affected vendor responses about the issues can be found in the associated individual vulnerability advisories.

NyxEngine – SDK references

Unicode support

Unicode support has been added to *NyxEngine*. However Unicode functions are not documented in this document because changes between function versions that use ASCII or UNICODE strings as input/output parameters are minor. Unicode functions are defined in the SDK and can be used normally. Such functions can be easily recognized by the appendix “W” which they have. For specific function definitions please refer to the SDK header files.

Nyx constants

```
#define NYX_ARCHIVE_ZIP 1
#define NYX_ARCHIVE_ZIP64 2
#define NYX_ARCHIVE_CAB 3
#define NYX_ARCHIVE_GZIP 4
#define NYX_ARCHIVE_RAR 5

#define NYX_VALIDATION_FILE_VALID 1
#define NYX_VALIDATION_FILE_BROKEN 2
#define NYX_VALIDATION_FILE_BROKEN_BUT_FIXABLE 3

#define NYX_SELECT_CURRENT_FILE -1
#define NYX_SELECT_PREVIOUS_FILE -2
#define NYX_SELECT_NEXT_FILE -3

#define NYX_FILESPAN_PRESENT 1
#define NYX_FILESPAN_NOT_PRESENT 0
#define NYX_FILESPAN_FROM_PREVIOUS_FILE -1
#define NYX_FILESPAN_TO_NEXT_FILE -2
#define NYX_FILESPAN_TO_NEXT_AND_PREVIOUS_FILE -3

#define NYX_RECOVER_ERROR -1
#define NYX_RECOVER_SUCCESS 0
#define NYX_RECOVER_NOT_SUPPORTED 1

#define NYX_VULN_FILENAME_TOO_LONG 1
#define NYX_VULN_SUSPICIOUS_COMPRESSION_RATIO 2
#define NYX_STEGO_FILENAME_UNPRINTABLE 3
#define NYX_VULN_FILENAME_TOO_SHORT 4
#define NYX_STEGO_SUSPICIOUS_UNPRINTABLE_DATA 5
#define NYX_STEGO_SUSPICIOUS_PRINTABLE_DATA 6
#define NYX_VULN_EXTRACT_VER_REQUIREMENT_SUSPICIOUS 7
#define NYX_CORRUPTION_CANNOT_ACCESS_DATA 8
#define NYX_CORRUPTION_CHECKSUM_MISMATCH 9
#define NYX_STEGO_UNREPORTED_FILE_FOUND 10
#define NYX_STEGO_FOUND_CLOAKED_DATA 11
#define NYX_VULN_INCORRECT_HEADER_DATA 12
#define NYX_VULN_CHECKSUM_NOT_SET 13
#define NYX_VULN_POSSIBLE_MULTIDISK_TAMPERING 14
#define NYX_VULN_FILE_WILL_EXECUTE_UPON_EXTRACTION 15
#define NYX_VULN_DUPLICATED_FILE_NAME 16

#define NYX_MAX_VULNERABILITIES 32
#define NYX_MAX_STEGANOGRAPHY 32
#define NYX_MAX_CORRUPTIONS 32
```

Nyx structures

```
typedef struct NYX_INSPECT_ENTRY{
            bool nyxDetectedVulnerability;
            bool nyxDetectedSteganography;
            bool nyxDetectedCorruptions;
            DWORD nyxReportedInfoId;
            DWORD nyxReportedRichInfoId;
            DWORD64 nyxReportedDataStart;
            DWORD nyxReportedDataSize;
            bool nyxIsDetectedAnomalyFixable;
}NYX_INSPECT_ENTRY, *PNYX_INSPECT_ENTRY;

typedef struct NYX_FILE_ARCHIVE_INFORMATION{
            DWORD nyxArchiveType;
            DWORD nyxTotalDiskNumber;
            DWORD nyxCurrentDiskNumber;
            DWORD nyxNumberOfFilesInCurrentDisk;
            DWORD nyxTotalNumberOfFilesInAllDisks;
            bool nyxMultiDiskInformationPresent;
            bool nyxDetectedFileIsInOverlay;
            DWORD64 nyxArchiveHeaderLocation;
            DWORD64 nyxArchiveHeaderStart;
}NYX_FILE_ARCHIVE_INFORMATION, *PNYX_FILE_ARCHIVE_INFORMATION;

typedef struct NYX_INFO_ENTRY{
            DWORD nyxReportedInfoId;
            DWORD nyxReportedRichInfoId;
            DWORD64 nyxReportedDataStart;
            DWORD nyxReportedDataSize;
            bool nyxIsDetectedAnomalyFixable;
}NYX_INFO_ENTRY, *PNYX_INFO_ENTRY;

typedef struct NYX_FILE_INFO{
            DWORD nyxCurrentFileId;
            DWORD64 nyxPackedContentHeader;
            DWORD64 nyxPackedContentLocation;
            DWORD64 nyxPackedContentSize;
            bool nyxFileNameIsUTF;
            bool nyxFileNameUnicodePresent;            //Reserved
            char nyxFileName[MAX_PATH];
            wchar_t nyxFileNameUnicode[MAX_PATH];      //Reserved
            DWORD nyxEncryptionAlgorithm;              //Reserved
            DWORD nyxCompressionAlgorithm;             //Reserved
            DWORD nyxCompressedSize;
            DWORD nyxUncompressedSize;
            DWORD nyxFileSpanningInfo;                 //Reserved
            DWORD nyxFileStartsOnDiskID;
            DWORD nyxFileChecksum;
            DWORD nyxFileCreationTime;
            DWORD nyxFileCreationDate;
            DWORD nyxFileAttributes;
            bool nyxFileIsPasswordProtected;
            bool nyxDetectedVulnerability;
            bool nyxDetectedSteganography;
            bool nyxDetectedCorruptions;
            NYX_INFO_ENTRY nyxVulnerabilityIDs[NYX_MAX_VULNERABILITIES];
            NYX_INFO_ENTRY nyxSteganographyIDs[NYX_MAX_STEGANOGRAPHY];
            NYX_INFO_ENTRY nyxCorruptionIDs[NYX_MAX_CORRUPTIONS];
}NYX_FILE_INFO, *PNYX_FILE_INFO;
```

NyxInitialize function

The **NyxInitialize** function is used to initialize the NyxEngine. The function is automatically called when the engine module is loaded and therefore its calling isn't necessary at this time.

Syntax

```
bool __stdcall NyxInitialize();
```

Parameters

None.

Return value

If initialization succeeds, *NyxInitializes* returns TRUE, otherwise, it returns FALSE.

Remarks

None.

Example

None.

NyxOpenArchive function

The **NyxOpenArchive** function is used to open a handle to archive. This function is always called before doing any actions with the selected file. Once all actions needed are performed with the archive opened handle to it should be close.

Syntax

```
long __stdcall NyxOpenArchive(
                  char* szFileName,
                  DWORD OpenMode
                  );
```

Parameters

szFileName
[in] Pointer to the full path of the file to open.

OpenMode
[in] Specifies needed type of access to file. This is GENERIC_READ, GENERIC_WRITE or a combination of both.

Return value

If initialization succeeds, *NyxOpenArchive* returns value other than INVALID_HANDLE_VALUE which is returned if the file can’t be located or opened. Returned value is a unique handle identifier for the opened archive and it’s compatible with Microsoft Windows functions that use file handles to perform actions.

Remarks

Specified constants are defined on MSDN page which refers to CreateFileA function.

Example

None.

NyxCloseArchive function

The **NyxCloseArchive** function is used to close an open archive handle.

Syntax

```
void __stdcall NyxCloseArchive(
                  long NyxFileHandle
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

Return value

None.

Remarks

Should be called for every archive open with the Nyx engine.

Example

None.

NyxIdentifyArchive function

The **NyxIdentifyArchive** function is used to check is the opened archive is supported by the Nyx engine.

Syntax

```
bool __stdcall NyxIdentifyArchive(
                 long NyxFileHandle
                 );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

Return value

If the archive is supported by the engine, *NyxIdentifyArchive* returns TRUE, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxSetFileSegment function

The **NyxSetFileSegment** function is used to set the range inside the selected file which is to be observed. This should be done right after opening a file and only if the program should inspect only one selected file slice and not the whole file.

Syntax

```
bool __stdcall NyxSetFileSegment(
                long NyxFileHandle,
                DWORD64 nyxSegmentStart,
                DWORD64 nyxSegmentSize
                );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.
nyxSegmentStart
[in] Selected start of the observed file segment.
nyxSegmentSize
[in] Selected size of the observed file segment.

Return value

NyxSetFileSegment returns TRUE if the file segment info was updated, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxGetArchiveName function

The **NyxGetArchiveName** function is used to retrieve the name of the identified archive type.

Syntax

```
char* __stdcall NyxGetArchiveName(
                  long NyxFileHandle
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

Return value

If the archive is supported by the engine, *NyxGetArchiveName* returns name of the identified archive type or NULL if the archive type is unknown.

Remarks

Current list of returned strings is: ZIP, ZIP64, RAR, CAB and GZIP. All upper case letters.

Example

None.

NyxGetArchiveProperties function

The **NyxGetArchiveProperties** function is used to retrieve additional information about the opened archive such as multiple disk information and total number of files present in the archive. However not all information is available for every archive type and this is due to archive file format specifics.

Syntax

```
bool __stdcall NyxGetArchiveProperties(
                  long NyxFileHandle,
                  void* NyxArchiveInfo
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

NyxArchiveInfo

[out] Pointer to NYX_FILE_ARCHIVE_INFORMATION structure which receives the additional archive information.

Return value

NyxGetArchiveProperties returns TRUE if the archive is supported by the engine and additional information was retrieved, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxGetNextFile function

The **NyxGetNextFile** function is used to retrieve information about the packed archive content. It reads the archive header in order to find information about the packed files. That information is returned by the function in a detailed structure. Calling this function sets the file cursor to the next file inside the archive.

Syntax

```
bool __stdcall NyxGetNextFile(
                  long NyxFileHandle,
                  void* NyxFileInfo
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

NyxArchiveInfo

[out] Pointer to NYX_FILE_INFO structure which receives the information about the next file inside the archive.

Return value

NyxGetNextFile returns TRUE if the archive is supported by the engine and file information was retrieved, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxGetPreviousFile function

The **NyxGetPreviousFile** function is used to retrieve information about the packed archive content. It reads the archive header in order to find information about the packed files. That information is returned by the function in a detailed structure. Calling this function sets the file cursor to the previous file inside the archive.

Syntax

```
bool __stdcall NyxGetPreviousFile(
                  long NyxFileHandle,
                  void* NyxFileInfo
                  );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.

NyxArchiveInfo
[out] Pointer to NYX_FILE_INFO structure which receives the information about the previous file inside the archive.

Return value

NyxGetPreviousFile returns TRUE if the archive is supported by the engine and file information was retrieved, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxGetSelectedFile function

The **NyxGetSelectedFile** function is used to retrieve information about the packed archive content. It reads the archive header in order to find information about the packed files. That information is returned by the function in a detailed structure. Calling this function sets the file cursor to the selected file inside the archive.

Syntax

```
bool __stdcall NyxGetSelectedFile(
                  long NyxFileHandle,
                  int NyxFileId,
                  void* NyxFileInfo
                  );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.

NyxFileId
[in] Selected file ID which is the file position inside the archive with the numbering starting from zero.

NyxArchiveInfo
[out] Pointer to NYX_FILE_INFO structure which receives the information about the selected file inside the archive.

Return value

NyxGetSelectedFile returns TRUE if the archive is supported by the engine and file information was retrieved, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxResetFileCurrsor function

The **NyxResetFileCurrsor** function is used to reset the file cursor to the first file inside the archive.

Syntax

```
bool __stdcall NyxResetFileCurrsor(
                 long NyxFileHandle
                 );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.

Return value

NyxResetFileCurrsor returns TRUE if the file cursor was reset, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxScanArchive function

The **NyxScanArchive** function is used to transverse all the files present in the archive. For each file present in the archive provided callback is called.

Syntax

```
void __stdcall NyxScanArchive(
                  long NyxFileHandle,
                  void* NyxReportCallBack
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

NyxReportCallBack

[in] Address of a callback function which is called for every file found inside the archive. A callback function receives two input parameters with the second being a pointer to NYX_FILE_INFO structure which holds the information about the selected file inside the archive.

```
typedef void(__stdcall *fNyxReportCallBack)(
            long nyxOpenFileHandle,
            void* NyxFileInfo
            );
```

Return value

None.

Remarks

None.

Example

None.

NyxInspectArchive function

The **NyxInspectArchive** function is used to do a detail archive content inspection in order to find vulnerabilities and steganography hidden data. Found data is reported one at the time and always just after it has been detected.

Syntax

```
void __stdcall NyxInspectArchive(
                  long NyxFileHandle,
                  void* NyxReportCallBack
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

NyxReportCallBack

[in] Address of a callback function which is called for every file found inside the archive. A callback function receives two input parameters with the second being a pointer to NYX_INSPECT_ENTRY structure which holds the information about the selected file inside the archive.

```
typedef void(__stdcall *fNyxReportCallBack)(
            long nyxOpenFileHandle,
            void* NyxInfoEntry
            );
```

Return value

None.

Remarks

None.

Example

None.

NyxValidateArchive function

The **NyxValidateArchive** function is used to validate the integrity of the selected archive which is inspected from the standpoint of data corruption integrity. These checks are purely superficial and the validation process only inspects the header data and the compressed items presence.

Syntax

```
bool __stdcall NyxValidateArchive(
                  long NyxFileHandle,
                  LPDWORD NyxArchiveValidation
                  );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.

NyxFileHandle
[in] Pointer to a DWORD variable that can hold the following values upon file inspection: NYX_VALIDATION_FILE_VALID, NYX_VALIDATION_FILE_BROKEN or NYX_VALIDATION_FILE_BROKEN_BUT_FIXABLE.

Return value

NyxValidateArchive returns TRUE if the file is valid, otherwise it returns FALSE.

Remarks

None.

Example

None.

NyxGetReportedIssueDescription function

The **NyxGetReportedIssueDescription** function is used to return a meaningful description of a detected archive format issue.

Syntax

```
char* __stdcall NyxGetReportedIssueDescription(
                  int NyxEncodedReportID
                  );
```

Parameters

NyxEncodedReportID

[in] Encoded ID of a detected vulnerability, steganography or corruption.

Return value

NyxGetReportedIssueDescription returns a detected vulnerability, steganography or corruption description.

Remarks

None.

Example

None.

NyxGetReportedIssueRichDescription function

The **NyxGetReportedIssueRichDescription** function is used to return a meaningful rich description of a detected archive format issue. Rich descriptions contain description of a detected issue and/or an ID of a vulnerability document that describes them.

Syntax

```
char* __stdcall NyxGetReportedIssueRichDescription(
                  int NyxEncodedReportRichID
                  );
```

Parameters

NyxEncodedReportRichID
[in] Encoded ID of a detected vulnerability, steganography or corruption.

Return value

NyxGetReportedIssueRichDescription returns a detected vulnerability, steganography or corruption rich description.

Remarks

None.

Example

None.

NyxExtractFileSlice function

The **NyxExtractFileSlice** function is used to copy the selected file part to the specified file on the disk.

Syntax

```
bool __stdcall NyxExtractFileSlice(
                  long NyxFileHandle,
                  char* szExtractedFile,
                  DWORD nyxDataStart,
                  DWORD nyxDataSize
                  );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.
szExtractedFile
[in] Path to the file to which will be selected file slice will be copied to.
nyxDataStart
[in] Start offset from which the selected file part will be copied.
nyxDataSize
[in] Size of the selected file part to copy.

Return value

NyxExtractFileSlice returns TRUE if the file slice was successfully extracted, otherwise it returns FALSE.

Remarks

If specified file exists it is overwritten.

Example

None.

NyxRecoverFile function

The **NyxRecoverFile** function is used to recover broken files from supported archive formats. If recovery is possible newly created archive will be an archive of the same type containing only the file whose recovery was attempted. Currently file recovery is available for: ZIP and RAR archive formats.

Syntax

```
long __stdcall NyxRecoverFile(
                  long NyxFileHandle,
                  int NyxFileId,
                  char* szExtractedFile
                  );
```

Parameters

NyxFileHandle

[in] Unique handle of the opened archive.

NyxFileId

[in] Selected file ID which is the file position inside the archive with the numbering starting from zero. If you are using this function parallel to NyxGetNextFile you can use the following constants instead of file ID: NYX_SELECT_CURRENT_FILE, NYX_SELECT_PREVIOUS_FILE or NYX_SELECT_NEXT_FILE.

szExtractedFile

[in] Path to the file to which will be selected file will be recovered to.

Return value

NyxRecoverFile attempts file recovery, estimates its success and returns one of the following values: NYX_RECOVER_ERROR, NYX_RECOVER_SUCCESS or NYX_RECOVER_NOT_SUPPORTED.

Remarks

If specified file exists it is overwritten.

Example

None.

NyxRecoverFileEx function

The **NyxRecoverFileEx** function is used to recover broken files from supported archive formats. If recovery is possible newly created archive will be an archive of the same type containing only the file whose recovery was attempted. Currently file recovery is available for: ZIP and RAR archive formats.

Syntax

```
long __stdcall NyxRecoverFileEx(
                  long NyxFileHandle,
                  DWORD nyxSegmentStart,
                  DWORD nyxSegmentSize,
                  char* szExtractedFile
                  );
```

Parameters

NyxFileHandle
[in] Unique handle of the opened archive.

nyxSegmentStart
[in] Indicates the start of the local file header of the file whose recovery will be attempted.

nyxSegmentSize
[in] Indicates the size of the local file header and compressed content whose recovery will be attempted.

szExtractedFile
[in] Path to the file to which will be selected file will be recovered to.

Return value

NyxRecoverFileEx attempts file recovery, estimates its success and returns one of the following values: NYX_RECOVER_ERROR, NYX_RECOVER_SUCCESS or NYX_RECOVER_NOT_SUPPORTED.

Remarks

If specified file exists it is overwritten.

Example

None.

END USER LICENSE AGREEMENT

Please read this End User License Agreement ("**EULA**") carefully and be sure you understand it. This EULA is a legal agreement between you (either an individual or a single entity) and ReversingLabs Corporation (“**ReversingLabs**”). You must review and either accept or reject the terms of this EULA before installing or using this Software Application. Clicking the "I ACCEPT" button below is just like signing a contract written on paper. By clicking the “I ACCEPT” button or installing or using the Software Application, you acknowledge that you have read all of the terms and conditions of this EULA, understand them, and agree to be legally bound by them.

If you have entered into a separate agreement with ReversingLabs permitting you to use the Software Application, that agreement, rather than this EULA, shall govern your use of the Software Application. If all or some portion of the Software Application has been licensed by another party for your use (e.g., your employer or an individual or company with which you conduct business), your right to use the Software Application is subject to the terms and conditions of the agreement(s) between ReversingLabs and the other party (the "**Master Software License and Services Agreement**"), whether the other party has agreed to the terms of this EULA or to the terms of the Master Software License and Services Agreement.

1. **DEFINITIONS**. The following capitalized terms used in this EULA have the meanings indicated. Additional terms are defined in context elsewhere in this EULA.
   a. **Computer**. The term “**Computer**” means a personal computer or personal computing device.
   b. **Delivery Date**. The term "**Delivery Date**" means (i) in the case of the Software Application that utilizes an activation key, the date on which ReversingLabs sends or otherwise makes available to you the activation key(s) for the Software Application or a method for creating them, and (ii) in the case of the Software Application that does not utilize an activation key, the date on which ReversingLabs sends you medium containing the Software Application, or otherwise provides you with a digital file containing the Software Application.
   c. **Documentation**. The term "**Documentation**" means all user manuals and training materials, whether provided in print or via electronic media, which relate to the Software Application.
   d. **Intellectual Property Rights**. The term "**Intellectual Property Rights**" means any rights or interests (whether owned by or licensed to a party), including but not limited to, those rights existing under common law or other non-statutory rights to the inventor or creator, now or in the future, under patent law, copyright law, trademark and unfair competition law, data and database protection law, trade secret law, and any and all similar proprietary rights or other rights to exclude. The term "Intellectual Property Rights" means such rights as they exist as of the date on which you agree to be bound by this EULA, and all such rights subsequently acquired by a party, to the extent necessary to perform under this Agreement.
   e. **Object Code**. The term "**Object Code**" means the machine readable code produced by a compiler. By way of clarification, and not limitation, the term "Object Code" means the source code transformed by compiler into a low-level language so that a machine can properly read and execute the code's underlying instructions.
   f. **Party** or **Parties**. the term "**Party**" or "**Parties**" means: (i) ReversingLabs, (ii) you; or (iii) ReversingLabs and you, as the context permits.
   g. **Software Application**. The term "**Software Application**" means the software accompanying this EULA, in Object Code, together with any of the following that may form a part of it or subsequently be provided by ReversingLabs for use with it: (i) Documentation, (ii) online services; and (iii) Updates.
   h. **Source Code**. The term "**Source Code**" means the computer programs, instructions and related material written in a human-readable source language in a form capable of serving as the input to a compiler or assembler program, and in a form capable of being modified, supported and enhanced by programmers familiar with the source language. By way of clarification, and not limitation, the term "Source Code" means the preferred form of the code for making modifications to it, including all modules it contains, plus any associated interface definition files, and scripts used to control compilation and installation of an executable. Unless otherwise agreed in writing, Source Code shall contain reasonable programmers' comments.

i. **Support Services**. The term "**Support Services**" means the software maintenance services, support services (including deployment support services) and any other services ReversingLabs may provide to users of the Software Application and that is governed by a separate agreement.

j. **Updates**. The term "**Updates**" means bug fixes, patches, improvements, new versions, new releases, or other revisions to or modifications of the Software Application that ReversingLabs may provide to you, including those it makes generally available to customers that receives the Support Services.

2. **Ownership of Software Application**. The Software Application is licensed, and not sold, to you. The Software Application is the property of ReversingLabs and its licensors (if any). Title to each copy of the Software Application and all related Intellectual Property Rights embodied therein or represented thereby shall remain at all times with ReversingLabs and/or its licensors, as shall all other rights not explicitly granted to you pursuant to this EULA. Any Third Party Software of which ReversingLabs is an authorized reseller may be accompanied by a separate license agreement, in which case that agreement, rather than this EULA, governs your use of the Third Party Software.

3. **Grant of Rights**.

   a. **Grant of Rights to You in Software Application**. In consideration of you fulfilling your obligations under this EULA, ReversingLabs, under its Intellectual Property Rights, hereby grants to you a limited, non-exclusive, non-sublicensable license to install one (1) copy of the Software Application on a computer owned, leased or otherwise controlled by you or your employer in Object Code only and solely for your or your employer's internal business purposes. You may make one (1) copy of the Software Application solely for backup or archival purposes.

   b. **Evaluation Software**. If ReversingLabs provides Software to you for evaluation use, then your rights are limited as described in this section. You may use the Software (and any Services ReversingLabs chooses to provide you in connection with it) in a manner consistent with the terms of this EULA solely for evaluation purposes for 90 days from the Delivery Date, or for such other period as may be indicated in writing by ReversingLabs at or after the time of delivery. In light of the fact that evaluation Software is provided to you free of charge, ReversingLabs disclaims the limited warranty set forth below in Section 10, and neither ReversingLabs nor any Released Party will be liable for direct damages related to evaluation Software, as explained more fully in Section 11(b). Evaluation copies of Software may include a "time-out" mechanism that will automatically downgrade the Software to trial-level Software or disable use of the Software and prevent access to Services at the end of the evaluation period.

   c. **No Implied Rights**. Nothing in this EULA shall be construed so as to grant you any rights other than those expressly provided herein. Any rights granted under this EULA to you must be expressly provided herein, and there shall be no implied rights pursuant to this EULA, based on any course of conduct or other construction or interpretation thereof. All rights and licenses not expressly granted herein are reserved.

4. **Restrictions and Limitations**. You acknowledge and agree that you shall comply with the following restrictions and limitations, and you acknowledge and agree that you shall not permit others to violate them:

   a. **Copying, Distribution and Use**. You shall not copy the Software Application, except in connection with installation of it as provided in Section 3.a (Grant of Rights to You in Software Application). You shall not sell, rent, lease, sublicense or redistribute the Software Application or its activation key(s) (if any), or use or permit others to install or directly or indirectly access or use the Software Application, its functionality, or its activation key(s) (if any).

   b. **Proprietary Notices**. You shall not alter or remove any copyright, trademark, patent, or other protective notices contained in or on the Software Application.

   c. **Reverse Engineering, Decompilation and Disassembly**. You shall not reverse engineer, decompile, or disassemble the Software Application or otherwise attempt to derive any related Source Code, except and only to the extent that any of these activities is permitted by applicable law despite this restriction. To the extent that the right to decompile, disassemble, or reverse engineer the Software Application is permitted by applicable law, you agree not to do so if ReversingLabs makes available to you a separate software module that allows you to achieve interoperability of an independently created computer program for use with the Software Application. You agree that, prior to attempting to achieve such interoperability, you shall obtain written notification from ReversingLabs that it is unwilling to make such a software module available within a reasonable period of time.

   d. **Modifications and Derivative Works**. You shall not modify or create derivative works of the Software Application.
   e. **Interference with Certain Features**. You shall not modify, disable, circumvent, avoid, bypass, remove, deactivate, impair or otherwise interfere with features of the Software Application that (i) enforce license restrictions or limits, or (ii) that reports technical or statistical information regarding the Software Application or its use to ReversingLabs.
   f. **Use of Prior Versions**. You shall not continue to use prior versions of the Software Application after installing an Update that wholly replaces the Software Application.
   g. **Activation Keys**. The Software Application's activation key (if any) is intended solely for your use. You are solely responsible for maintaining the confidentiality and security of your activation key(s). You are solely responsible and liable for any and all use of your activation key(s) and for activities that occur on or through your activation key(s). You agree to notify ReversingLabs immediately about any unauthorized access to, or use of, any of your activation key(s). The Software Application contains technological measures designed to prevent its unlicensed or illegal use. You agree that ReversingLabs may use those measures in its sole discretion.
5. **Maintenance and Support**. Technical support for the Software Application may be found in the Help menu within the Software Application or in the Documentation (if provided). Unless you subscribe to ReversingLabs' Support Services, you are not entitled to receive additional maintenance or support for the Software Application (though any Updates ReversingLabs may provide you will be covered by this EULA, unless ReversingLabs requires you to accept a new agreement at the time they are provided). If you with to engage ReversingLabs for Support Services, ReversingLabs shall provide you with a separate agreement, which terms shall govern your receipt of, and ReversingLabs' provision of, Support Services. Whether or not you subscribe to the Support Services, ReversingLabs reserves the right to provide you with updates or supplements to the Software Application when ReversingLabs considers it necessary to do so to ensure that the Software Application functions properly.
6. **Warranty Disclaimer**. REVERSINGLABS AND ITS LICENSORS AND LICENSORS' DISTRIBUTORS DISCLAIM ALL WARRANTIES WITH RESPECT TO THE SOFTWARE APPLICATION AND ALL THIRD PARTY PRODUCTS OR SERVICES YOU MAY UTILIZE IN CONNECTION WITH THE SOFTWARE APPLICATION, WHETHER EXPRESS OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN PARTICULAR, REVERSINGLABS DOES NOT REPRESENT THAT THE SOFTWARE APPLICATION IS ERROR FREE, WILL OPERATE IN AN UNINTERRUPTED MANNER, IS COMPLETELY SECURE, OR WILL INTEROPERATE WITH THIRD PARTY SOFTWARE OR SERVICES. UNLESS YOU HAVE SUBSCRIBED TO THE SUPPORT SERVICES, WHICH GUARANTEES A PARTICULAR LEVEL OF SERVICE AND/OR A FIXED TERM OF SERVICE (AS EXPLICITLY SPECIFIED IN A SEPARATE AGREEMENT), ALL SERVICES THAT REVERSINGLABS MAY PROVIDE IS DONE SO ON AN "AS IS" AND "AS AVAILABLE" BASIS AND ARE SUBJECT TO CHANGE OR TERMINATION AT ANY TIME AND FOR ANY REASON WITHOUT NOTICE.
7. **Exclusion of Damages; Limitation of Liability**.
   a. **Consequential Damages Waiver**. TO THE MAXIMUM EXTENT PERMITTED BY LAW (INCLUDING ANY APPLICABLE CONSUMER PROTECTION LAW OF A FOREIGN JURISDICTION), NEITHER REVERSINGLABS NOR ANY OF ITS DIRECTORS, OFFICERS, EMPLOYEES, CONTROLLED OR CONTROLLING ENTITIES, LICENSORS OR LICENSORS' DISTRIBUTORS (EACH, A "**RELEASED PARTY**"), WILL HAVE ANY LIABILITY TO YOU FOR INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES (INCLUDING, WITHOUT LIMITATION, ANY LOSS OF USE, LOST PROFITS, BUSINESS OR REVENUE, LOSS OF GOODWILL OR OTHER ECONOMIC ADVANTAGE, OR LOSS OF PRIVACY) ARISING OUT OF OR RELATED TO THIS EULA, THE SOFTWARE APPLICATION, EVEN IF REVERSINGLABS OR A RELEASED PARTY HAS BEEN ADVISED OF, OR KNEW OR SHOULD HAVE KNOWN OF, THE POSSIBILITY OF SUCH DAMAGES.
   b. **Limitation of Liability**. IN NO EVENT WILL REVERSINGLABS' AND THE RELEASED PARTIES' TOTAL LIABILITY WITH RESPECT TO ALL CLAIMS ARISING OUT OF OR RELATED TO THIS EULA, THE SOFTWARE APPLICATION (INCLUDING CLAIMS OF NEGLIGENCE AND STRICT LIABILITY) EXCEED THE LOWER OF (i) THE AGGREGATE DIRECT DAMAGES ACTUALLY INCURRED BY YOU, OR (ii) US $5.00.
   c. **Exceptions**. SOME JURISDICTIONS LIMIT THE EXCLUSION OF DAMAGES OR LIMITATION OF LIABILITY, SO THE ABOVE EXCLUSIONS AND LIMITATIONS MAY NOT APPLY TO YOU. IF ANY PART OF THE EXCLUSIONS OF DAMAGES OR LIMITATIONS OF LIABILITY SET FORTH IN THIS EULA IS UNENFORCEABLE UNDER APPLICABLE

LAW, REVERSINGLABS' AND THE RELEASED PARTIES' AGGREGATE LIABILITY WILL BE LIMITED TO THE MAXIMUM EXTENT PERMITTED BY LAW, EVEN IF ANY REMEDY FAILS ITS ESSENTIAL PURPOSE.

8. **Term and Termination**. The term of this EULA shall commence upon your installation or use of the Software Application and continue indefinitely, unless you and ReversingLabs enter into a new agreement that entirely replaces this EULA or ReversingLabs terminates this EULA as provided herein. Without prejudice to any other rights, ReversingLabs may, at its option and in its sole discretion, terminate this EULA if you fail to comply with its terms and conditions, or for any other reason, including, but not limited to, ReversingLabs' convenience. This EULA terminates immediately upon termination of the Master Software License and Services Agreement. If ReversingLabs terminates this EULA or the Master Software License and Services Agreement, you shall immediately stop using the Software Application and destroy all copies of the Software Application and all of its component parts. The Parties' respective rights and obligations under Sections 2 (Ownership of Software Application), 4 (Restrictions and Limitations), 6 (Limited Warranties and Warranty Disclaimer), 7 (Exclusion of Damages; Limitation of Liability); and Section 9 (General Provisions) shall survive the termination of this EULA.
9. **General Provisions**.
    a. **Compliance with Laws; Export and Import Restrictions**. You shall comply with all applicable laws and regulations of governmental bodies and agencies in your use of the Software Application. In particular, you acknowledge and agree that the Software Application is of United States origin, is subject to United States export laws and regulations, and that you shall not allow the Software Application to be exported or re-exported: (i) to countries subject to US embargo (currently Cuba, Iran, Libya, North Korea, Sudan and Syria), (ii) to persons or entities on: (a) the US Commerce Department Denied Persons List (currently accessible at http://www.bxa.doc.gov/DPL/Default.shtm), (b) the Entity List (currently accessible at http://www.bxa.doc.gov/Entities/); or (c) the US Treasury Department list of Specially Designated Nationals and Blocked Persons (currently accessible at http://www.ustreas.gov/ofac/t11sdn.pdf); or (iii) to persons or entities involved with missile technology or nuclear, chemical or biological weapons. The Software Application may also be subject to the export, import or other laws of other jurisdictions. You shall not use or transfer the Software Application in violation of any US or foreign laws or regulations, or permit any other party to do so.
    b. **Waiver; Severability**. No delay or omission by either party to exercise any right or power arising upon the other Party's nonperformance or breach will impair that right or power or be construed as a waiver of it. Any waiver must be in writing and signed by the waiving Party. A waiver on one occasion will not be construed as a waiver of any subsequent event of nonperformance or breach. If any provision of this EULA is declared to be unenforceable for any reason, the remainder of this EULA will continue in full force and effect, and the unenforceable provision will be deemed modified to the extent necessary to comply with the applicable requirements of law, while retaining to the maximum extent permitted by law its intended effect, scope and economic effect.
    c. **Governing Law; Dispute Resolution**. The interpretation and performance of this EULA shall be governed by the laws of the Commonwealth of Massachusetts, USA, applicable to contracts executed in and performed entirely within Massachusetts, but excluding any choice of law principles that would result in the application of the laws of another jurisdiction. The Parties expressly agree that the United Nations Convention on Contracts for the International Sale of Goods shall not apply to this EULA. Any dispute arising under or related to this EULA shall be resolved exclusively by the state or federal courts sitting in the judicial district that includes Boston, Massachusetts, and both parties submit to personal jurisdiction and venue before such courts.
    d. **Software Application and EULA Transfer**. This EULA and the Software Application may be transferred to another party without ReversingLabs' consent once by the original licensee. Subsequent transfers require ReversingLabs' prior written consent.
    e. **Entire Agreement**. This EULA sets forth the entire agreement between you and ReversingLabs with respect to its subject matter, and it supersedes all prior communications, understandings and agreements, as well as the terms and conditions set forth in or on any purchase order, acknowledgement form, check, or any other document or instrument you may issue to ReversingLabs or transmit in connection with any payment for the Software Application.