RBS-2019-009

cliEncode ActiveX Control
Multiple Methods Handling Stack Buffer Overflows

Vulnerable Program Details

Details for tested products and versions:

Vendor:	INITECH Co., Ltd.
Product:	cliEncode ActiveX Control (cliEncode.ocx)
Version:	2.2.1.1

NOTE: Other versions than the one listed above are likely affected.

Credits

Carsten Eiram, Risk Based Security
Twitter: @RiskBased

Impact

The cliEncode ActiveX control (cliEncode.ocx) contains multiple stack-based buffer overflows that may allow an attacker to compromise a user's system.

Vulnerability Details

CliEncrypt() Method Handling Stack Buffer Overflow

The CliEncrypt() method accepts a single argument as defined below:

```
[id(0x00000001)]
BSTR CliEncrypt(BSTR Msg);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied "Msg" argument into a 512-byte stack buffer.

```
.text:10001989                mov     esi, [esp+6E4h+bstrMsg]
.text:10001990                or      ecx, 0FFFFFFFFh
.text:10001993                mov     edi, esi
.text:10001995                xor     eax, eax
.text:10001997                add     esp, 8
.text:1000199A                xor     ebx, ebx
.text:1000199C                repne scasb
.text:1000199E                not     ecx
.text:100019A0                dec     ecx
```

```
.text:100019A1                  lea     edi, [esp+6DCh+var_40C]
.text:100019A8                  mov     edx, ecx
.text:100019AA                  mov     ecx, 100h
.text:100019AF                  rep stosd
.text:100019B1                  mov     ecx, 80h
.text:100019B6                  lea     edi, [esp+6DCh+szDest] ; char[512]
.text:100019BD                  rep stosd
.text:100019BF                  mov     ecx, edx
.text:100019C1                  lea     edi, [esp+6DCh+szDest] ; char[512]
.text:100019C8                  mov     eax, ecx
.text:100019CA                  shr     ecx, 2
.text:100019CD                  rep movsd
.text:100019CF                  mov     ecx, eax
.text:100019D1                  and     ecx, 3
.text:100019D4                  test    edx, edx
.text:100019D6                  rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

CliDecrypt() Method Handling Stack Buffer Overflow

The CliDecrypt() method accepts a single argument and is defined as follows:

```
[id(0x00000002)]
BSTR CliDecrypt(BSTR Msg);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied “Msg” argument into a 1024-byte stack buffer.

```
.text:10001B69                  mov     esi, [esp+6E4h+bstrMsg]
.text:10001B70                  or      ecx, 0FFFFFFFFh
.text:10001B73                  mov     edi, esi
.text:10001B75                  xor     eax, eax
.text:10001B77                  add     esp, 8
.text:10001B7A                  repne scasb
.text:10001B7C                  not     ecx
.text:10001B7E                  dec     ecx
.text:10001B7F                  lea     edi, [esp+6DCh+szBuf1] ; char[512]
.text:10001B86                  mov     ebx, ecx
.text:10001B88                  mov     ecx, 80h
.text:10001B8D                  rep stosd
.text:10001B8F                  mov     ecx, 100h
.text:10001B94                  lea     edi, [esp+6DCh+szDest] ; char[1024]
.text:10001B9B                  rep stosd
.text:10001B9D                  mov     ecx, ebx
.text:10001B9F                  lea     edi, [esp+6DCh+szDest] ; char[1024]
.text:10001BA6                  mov     edx, ecx
.text:10001BA8                  shr     ecx, 2
.text:10001BAB                  rep movsd
.text:10001BAD                  mov     ecx, edx
.text:10001BAF                  and     ecx, 3
.text:10001BB2                  rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

FileEncrypt() Method Handling Stack Buffer Overflow

The FileEncrypt() method accepts a single argument and is defined as follows:

```
[id(0x00000003)]
VARIANT_BOOL FileEncrypt(BSTR FileName);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied “FileName” argument into a 128-byte stack buffer.

```
.text:10001D28                mov     edi, [esp+21Ch+bstrFileName]
.text:10001D2F                or      ecx, 0FFFFFFFFh
.text:10001D32                xor     eax, eax
.text:10001D34                add     esp, 8
.text:10001D37                repne scasb
.text:10001D39                not     ecx
.text:10001D3B                sub     edi, ecx
.text:10001D3D                lea     edx, [esp+214h+szDest] ; char[128]
.text:10001D44                mov     eax, ecx
.text:10001D46                mov     esi, edi
.text:10001D48                mov     edi, edx
.text:10001D4A                shr     ecx, 2
.text:10001D4D                rep movsd
.text:10001D4F                mov     ecx, eax
.text:10001D51                xor     eax, eax
.text:10001D53                and     ecx, 3
.text:10001D56                rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

FileDecrypt() Method Handling Stack Buffer Overflow

The FileDecrypt() method accepts a single argument and is defined as follows:

```
[id(0x00000004)]
VARIANT_BOOL FileDecrypt(BSTR FileName);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied “FileName” argument into a 128-byte stack buffer.

```
.text:10001FE3                mov     edi, [esp+224h+strFileName]
.text:10001FEA                lea     eax, [esp+224h+Delim]
.text:10001FEE                push    eax             ; Delim
.text:10001FEF                push    edi             ; Str
.text:10001FF0                call    ds:strtok
```

```
.text:10001FF6                  mov     edx, eax
.text:10001FF8                  or      ecx, 0FFFFFFFFh
.text:10001FFB                  xor     eax, eax
.text:10001FFD                  lea     ebx, [esp+22Ch+szFilename] ; char[128]
.text:10002004                  repne scasb
.text:10002006                  not     ecx
.text:10002008                  sub     edi, ecx
.text:1000200A                  push    offset aDec     ; "dec"
.text:1000200F                  mov     eax, ecx
.text:10002011                  mov     esi, edi
.text:10002013                  mov     edi, ebx
.text:10002015                  push    '.'
.text:10002017                  shr     ecx, 2
.text:1000201A                  rep movsd
.text:1000201C                  mov     ecx, eax
.text:1000201E                  push    edx
.text:1000201F                  and     ecx, 3
.text:10002022                  push    offset aSCS     ; "%s%c%s"
.text:10002027                  rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

FileRestore() Method Handling Stack Buffer Overflow

The FileRestore() method accepts a single argument and is defined as follows:

```
[id(0x00000005)]
VARIANT_BOOL FileRestore(BSTR FileName);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied “FileName” argument into a 128-byte stack buffer.

```
.text:10002150 M_FileRestore   proc near               ; DATA XREF: .rdata:10007608o
.text:10002150
.text:10002150 szFilename      = byte ptr -300h        ; char[128]
.text:10002150 szDest          = byte ptr -280h        ; char[128]
.text:10002150 szDstBuf        = byte ptr -200h        ; char[512]
.text:10002150 bstrFileName    = dword ptr  4
.text:10002150
.text:10002150                  sub     esp, 300h
.text:10002156                  or      ecx, 0FFFFFFFFh
.text:10002159                  xor     eax, eax
.text:1000215B                  lea     edx, [esp+300h+szFilename] ; char[128]
.text:1000215F                  push    esi
.text:10002160                  push    edi
.text:10002161                  mov     edi, [esp+308h+bstrFileName]
.text:10002168                  repne scasb
.text:1000216A                  not     ecx
.text:1000216C                  sub     edi, ecx
.text:1000216E                  mov     eax, ecx
.text:10002170                  mov     esi, edi
```

```
.text:10002172                  mov     edi, edx
.text:10002174                  shr     ecx, 2
.text:10002177                  rep movsd
.text:10002179                  mov     ecx, eax
.text:1000217B                  xor     eax, eax
.text:1000217D                  and     ecx, 3
.text:10002180                  rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

CertCopy() Method Handling Stack Buffer Overflow

The CertCopy() method accepts two arguments and is defined as follows:

```
[id(0x00000009)]
BSTR CertCopy(
                short type,
                BSTR cn);
```

When the method is called, the function responsible for handling it in cliEncode.ocx eventually copies the supplied “cn” argument into a 1024-byte stack buffer.

```
.text:10002955                  mov     edi, [esp+534h+bstrCn]
.text:1000295C                  or      ecx, 0FFFFFFFFh
.text:1000295F                  repne scasb
.text:10002961                  not     ecx
.text:10002963                  lea     esi, [esp+534h+szBuf] ; char[128]
.text:1000296A                  sub     edi, ecx
.text:1000296C                  mov     eax, ecx
.text:1000296E                  mov     [esp+534h+pszBuf], esi
.text:10002972                  mov     esi, edi
.text:10002974                  mov     edi, [esp+534h+pszBuf]
.text:10002978                  shr     ecx, 2
.text:1000297B                  rep movsd
.text:1000297D                  mov     ecx, eax
.text:1000297F                  and     ecx, 3
.text:10002982                  cmp     [esp+534h+type], 1
.text:1000298B                  rep movsb
```

As no bounds checks are performed, this may lead to a stack-based buffer overflow and code execution.

Solution

The vendor has deprecated the ActiveX control, and KrCERT/CC plans to set the kill-bit.

References

RBS:	RBS-2019-009[1]
VulnDB:	202028, 202029, 202030, 202031, 202032, 202033, 202034

Timeline

2019-01-17	Vulnerabilities discovered.
2019-02-01	Vulnerabilities reported to KrCERT/CC.
2019-04-04	Alerts published to VulnDB customers.
2019-05-21	Publication of this vulnerability report.

[1] https://www.riskbasedsecurity.com/research/RBS-2019-009.pdf

About Risk Based Security

Risk Based Security offers clients fully integrated security solutions, combining real-time vulnerability and threat data, as well as the analytical resources to understand the implications of the data, resulting in not just security, but the right security.

Company History

Risk Based Security, Inc. (RBS) was established to support organizations with the technology to turn security data into actionable information and a competitive advantage. We do so by enhancing the research available and providing a first of its kind risk identification and evidence-based security management service.

As a data driven and vendor neutral organization, RBS is able to deliver focused security solutions that are timely, cost effective, and built to address the specific threats and vulnerabilities most relevant to the organizations we serve. We not only maintain vulnerability and data breach databases, we also use this information to inform our entire practice.

Solutions

VulnDB - Vulnerability intelligence, alerting, and third party library tracking based on the largest and most comprehensive vulnerability database in the world. Available as feature-rich SaaS portal or powerful API. Vendor evaluations including our Vulnerability Timeline and Exposure Metrics (VTEM), Cost of Ownership ratings, Code Maturity, and Social Risk Scores.

Cyber Risk Analytics - Extensive data breach database including interactive dashboards and breach analytics. Clients are able to gather and analyze security threat and data breach information on businesses, industries, geographies, and causes of loss. It also allows monitoring of domains for data breaches and leaked credentials as well as implementing a continuous vendor management program with our PreBreach data.

YourCISO - Revolutionary service that provides organizations an affordable security solution including policies, vulnerability scans, awareness material, incident response, and access to high quality information security resources and consulting services.

Vulnerability Assessments (VA) and Pentesting - Regularly scheduled VAs and pentests help an organization identify weaknesses before the bad guys do. Managing the most comprehensive VDB puts us in a unique position to offer comprehensive assessments, combining the latest in scanning technology and our own data. Detailed and actionable reports are provided in a clear and easy to understand language.

Security Development Lifecycle (SDL) - Consulting, auditing, and verification specialized in breaking code, which in turn greatly increases the security of products.