RBS-2019-011

Innorix InnoFD6 ActiveX Control
Multiple Methods Handling Stack Buffer Overflows

Vulnerable Program Details

Details for tested products and versions:

Vendor:	Innorix
Product:	InnoFD6 ActiveX Control (InnoFD6.dll)
Version:	6.0.4.5150

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

Credits

Carsten Eiram, Risk Based Security
Twitter: @RiskBased

Impact

The InnoFD6 ActiveX control (InnoFD6.dll) contains multiple stack-based buffer overflows that may allow an attacker to compromise a user's system.

Vulnerability Details

DownloadAndOpen() Method Stack Buffer Overflow

The ActiveX control provides the DownloadAndOpen() method, which accepts one mandatory argument and three optional ones as defined below:

```
[id(0x0000025b), helpstring("¸Þ¼µå DownloadAndOpen")]
void DownloadAndOpen(
                [in] BSTR bstrURL,
                [in, optional, defaultvalue("")] BSTR bstrFilename,
                [in, optional, defaultvalue(0)] int64 nFilesize,
                [in, optional, defaultvalue(-1)] VARIANT_BOOL bOverwrite);
```

When the DownloadAndOpen() method is called, the function responsible for handling it in InnoFD6.dll eventually copies the "bstrURL" argument into a 2086 wide-character stack buffer via a call to lstrcpyW().

```
.text:10031BA6                mov     ebx, [ebp+14D0h+bstrURL]
.text:10031BAC                push    esi
.text:10031BAD                mov     esi, [ebp+14D0h+pThis]
```

```
.text:10031BB3                push    edi
.text:10031BB4                mov     [ebp+14D0h+var_14C4], eax
.text:10031BB7                call    Target
.text:10031BBD                movzx   ecx, byte ptr [esi+5C8h]
.text:10031BC4                push    ebx             ; lpString2
.text:10031BC5                lea     edx, [ebp+14D0h+wzURL] ; wchar[2086]
.text:10031BCB                push    edx             ; lpString1
.text:10031BCC                mov     [ebp+14D0h+CodePage], eax
.text:10031BCF                mov     [ebp+14D0h+var_14C8], ecx
.text:10031BD2                call    ds:lstrcpyW
```

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

SingleDownload() Method Stack Buffer Overflow

The ActiveX control provides the SingleDownload() method, which accepts one mandatory argument and two optional ones as defined below:

```
[id(0x0000025c), helpstring("¸Þ¼µå SingleDownload")]
void SingleDownload(
                [in] BSTR bstrURL,
                [in, optional, defaultvalue("")] BSTR bstrFilename,
                [in, optional, defaultvalue(0)] int64 nFilesize);
```

When the SingleDownload() method is called, the function responsible for handling it in InnoFD6.dll eventually copies the “bstrURL” argument into a 2086 wide-character stack buffer via a call to lstrcpyW().

```
.text:1001DA40                mov     ebx, [ebp+1960h+bstrURL]
.text:1001DA46                push    esi
.text:1001DA47                mov     esi, [ebp+1960h+arg_0]
.text:1001DA4D                push    edi
.text:1001DA4E                mov     edi, [ebp+1960h+arg_8]
.text:1001DA54                mov     [ebp+1960h+var_1954], ebx
.text:1001DA57                mov     [ebp+1960h+var_1958], edi
.text:1001DA5A                call    Target
.text:1001DA60                mov     [ebp+1960h+CodePage], eax
.text:1001DA63                push    ebx             ; lpString2
.text:1001DA64                lea     eax, [ebp+1960h+wzURL] ; wchar[2086]
.text:1001DA6A                push    eax             ; lpString1
.text:1001DA6B                call    ds:lstrcpyW
```

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

AppendFile() Method Stack Buffer Overflow

The ActiveX control provides the AppendFile() method, which accepts one mandatory argument and two optional ones as defined below:

```
[id(0x000001f5), helpstring("¸Þ¼µå AppendFile")]
void AppendFile(
                [in] BSTR bstrURL,
                [in, optional, defaultvalue("")] BSTR bstrFilename,
                [in, optional, defaultvalue(0)] int64 varFilesize);
```

When the AppendFile() method is called, the function responsible for handling it in InnoFD6.dll eventually copies the “bstrURL” argument into a 2086 wide-character stack buffer via a call to lstrcpyW().

```
.text:10031904                mov     edi, [ebp+1084h+bstrURL]
.text:1003190A                mov     ecx, esi
.text:1003190C                mov     [ebp+1084h+Src], eax
.text:1003190F                call    sub_1000B910
.text:10031914                test    al, al
.text:10031916                jnz     short loc_10031932
.text:10031932 loc_10031932:                           ; CODE XREF: M_AppendFile+46j
.text:10031932                call    Target
.text:10031938                mov     ecx, esi
.text:1003193A                mov     [ebp+1084h+CodePage], eax
.text:1003193D                call    sub_1000B910
.text:10031942                push    edi              ; lpString2
.text:10031943                lea     edx, [ebp+1084h+wzURL] ; wchar[2086]
.text:10031946                push    edx              ; lpString1
.text:10031947                call    ds:lstrcpyW
```

As no bounds checks are performed, this may result in a stack-based buffer overflow.

Later in another function the optional “bstrFilename” argument is processed if supplied. Here the filename is copied into a 260 wide-character stack buffer via a call to wcsncpy().

```
.text:10030BF5                push    eax              ; pszPath
.text:10030BF6                call    ebp ; PathFindFileNameW
.text:10030BF8                push    206h             ; size_t
.text:10030BFD                mov     esi, eax
.text:10030BFF                lea     eax, [esp+0F3Ch+wzFilename+2] ; wchar[260]
.text:10030C06                push    ebx              ; int
.text:10030C07                push    eax              ; void *
.text:10030C08                mov     [esp+0F44h+wzFilename], bx ; wchar[260]
.text:10030C10                call    _memset
.text:10030C15                mov     eax, esi
.text:10030C17                add     esp, 0Ch
.text:10030C1A                lea     edx, [eax+2]
.text:10030C1D                lea     ecx, [ecx+0]
.text:10030C20
.text:10030C20 loc_10030C20:                           ; CODE XREF: sub_10030A90+199j
.text:10030C20                mov     cx, [eax]
.text:10030C23                add     eax, 2
.text:10030C26                cmp     cx, bx
.text:10030C29                jnz     short loc_10030C20
.text:10030C2B                sub     eax, edx
.text:10030C2D                sar     eax, 1
.text:10030C2F                push    eax              ; size_t
```

```
.text:10030C30                  lea     ecx, [esp+0F3Ch+wzFilename] ; wchar[260]
.text:10030C37                  push    esi             ; wchar_t *
.text:10030C38                  push    ecx             ; wchar_t *
.text:10030C39                  call    _wcsncpy
```

As the supplied size argument to wcsncpy() is the length of the source string, no boundary checks are basically performed. This may result in a stack-based buffer overflow.

Solution

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

References

RBS:	RBS-2019-011[1]
VulnDB:	202035, 202036, 202037, 202038

Timeline

2019-01-29	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-2019-011.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.