RBS-2019-010

INITECH SCControl ActiveX Control
Multiple Methods Handling Stack Buffer Overflows

Vulnerable Program Details

Details for tested products and versions:

Vendor:	INITECH Co., Ltd.
Product:	SCControl ActiveX Control (SCControl.ocx)
Version:	2.0.0.2

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

Credits

Carsten Eiram, Risk Based Security
Twitter: @RiskBased

Impact

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

Vulnerability Details

GetOTPRes() Method Handling Stack Buffer Overflow

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

```
[id(0x00000005)]
BSTR GetOTPRes(BSTR Challenge);
```

When the method is called, the responsible function in SCControl.ocx eventually zeroes a 1024-byte stack buffer and then copies the "Challenge" argument passed to the method into it using an inline strcpy() call.

```
.text:10001A89                mov     ecx, 100h
.text:10001A8E                xor     eax, eax
.text:10001A90                lea     edi, [esp+82Ch+szDest] ; char[1024]
.text:10001A97                lea     edx, [esp+82Ch+szDest] ; char[1024]
.text:10001A9E                rep stosd
.text:10001AA0                mov     ecx, 100h
.text:10001AA5                lea     edi, [esp+82Ch+var_80C]
.text:10001AA9                rep stosd
```

```
.text:10001AAB                  mov     edi, [esp+82Ch+bstrChallenge]
.text:10001AB2                  or      ecx, 0FFFFFFFFh
.text:10001AB5                  repne scasb
.text:10001AB7                  not     ecx
.text:10001AB9                  sub     edi, ecx
.text:10001ABB                  push    offset aGetotpres ; "GetOTPRes"
.text:10001AC0                  mov     eax, ecx
.text:10001AC2                  mov     esi, edi
.text:10001AC4                  mov     edi, edx
.text:10001AC6                  mov     [esp+830h+var_4], 0
.text:10001AD1                  shr     ecx, 2
.text:10001AD4                  rep movsd
.text:10001AD6                  mov     ecx, eax
.text:10001AD8                  and     ecx, 3
.text:10001ADB                  rep movsb
```

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

CreateSession() Method Handling Stack Buffer Overflows

The CreateSession() method accepts three arguments as defined below:

```
[id(0x00000010)]
void CreateSession(
                BSTR SRandom,
                BSTR Crypto,
                long KeyNo);
```

When the method is called, the responsible function in SCControl.ocx eventually zeroes two 1024-byte stack buffers and then copies the ”Random” and “Crypto” arguments passed to the method into them using inline strcpy() calls.

```
.text:100029D2                  lea     edi, [esp+80Ch+szRandom] ; char[1024]
.text:100029D9                  rep stosd
.text:100029DB                  mov     ecx, 100h
.text:100029E0                  lea     edi, [esp+80Ch+szCrypto] ; char[1024]
.text:100029E4                  rep stosd
.text:100029E6                  mov     edi, [esp+80Ch+bstrSRandom]
.text:100029ED                  or      ecx, 0FFFFFFFFh
.text:100029F0                  repne scasb
.text:100029F2                  not     ecx
.text:100029F4                  sub     edi, ecx
.text:100029F6                  lea     ebx, [esp+80Ch+szRandom] ; char[1024]
.text:100029FD                  mov     eax, ecx
.text:100029FF                  mov     esi, edi
.text:10002A01                  mov     edi, ebx
.text:10002A03                  lea     ebx, [esp+80Ch+szCrypto] ; char[1024]
.text:10002A07                  shr     ecx, 2
.text:10002A0A                  rep movsd
.text:10002A0C                  mov     ecx, eax
.text:10002A0E                  xor     eax, eax
.text:10002A10                  and     ecx, 3
```

```
.text:10002A13                push    offset aCreatesession ; "CreateSession"
.text:10002A18                rep movsb
.text:10002A1A                mov     edi, [esp+810h+bstrCrypto]
.text:10002A21                or      ecx, 0FFFFFFFFh
.text:10002A24                repne scasb
.text:10002A26                not     ecx
.text:10002A28                sub     edi, ecx
.text:10002A2A                mov     eax, ecx
.text:10002A2C                mov     esi, edi
.text:10002A2E                mov     edi, ebx
.text:10002A30                shr     ecx, 2
.text:10002A33                rep movsd
.text:10002A35                mov     ecx, eax
.text:10002A37                and     ecx, 3
.text:10002A3A                rep movsb
```

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

Load() Method Handling Stack Buffer Overflows

The Load() method accepts two arguments as defined below:

```
[id(0x00000016)]
void Load(
                BSTR HostRandom,
                BSTR HostSign);
```

When the method is called, the responsible function in SCControl.ocx eventually zeroes two 512-byte stack buffers and then copies the ”HostRandom” and “HostSign” arguments passed to the method into them using inline strcpy() calls.

```
.text:1000300B                mov     ecx, 80h
.text:10003010                xor     eax, eax
.text:10003012                lea     edi, [esp+414h+szBuf1] ; char[512]
.text:10003019                lea     edx, [esp+414h+szBuf1] ; char[512]
.text:10003020                rep stosd
.text:10003022                mov     ecx, 80h
.text:10003027                lea     edi, [esp+414h+szBuf2] ; char[512]
.text:1000302B                rep stosd
.text:1000302D                mov     edi, [esp+414h+bstrHostRandom]
.text:10003034                or      ecx, 0FFFFFFFFh
.text:10003037                repne scasb
.text:10003039                not     ecx
.text:1000303B                sub     edi, ecx
.text:1000303D                mov     dword ptr [esp+414h+Data], 1
.text:10003045                mov     eax, ecx
.text:10003047                mov     esi, edi
.text:10003049                mov     edi, edx
.text:1000304B                lea     edx, [esp+414h+szBuf2] ; char[512]
.text:1000304F                shr     ecx, 2
.text:10003052                rep movsd
```

```
.text:10003054                  mov     ecx, eax
.text:10003056                  xor     eax, eax
.text:10003058                  and     ecx, 3
.text:1000305B                  rep movsb
.text:1000305D                  mov     edi, [esp+414h+bstrHostSign]
.text:10003064                  or      ecx, 0FFFFFFFFh
.text:10003067                  repne scasb
.text:10003069                  not     ecx
.text:1000306B                  sub     edi, ecx
.text:1000306D                  mov     eax, ecx
.text:1000306F                  mov     esi, edi
.text:10003071                  mov     edi, edx
.text:10003073                  shr     ecx, 2
.text:10003076                  rep movsd
.text:10003078                  mov     ecx, eax
.text:1000307A                  and     ecx, 3
.text:1000307D                  rep movsb
```

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

Mac() Method Handling Stack Buffer Overflow

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

```
[id(0x00000006)]
BSTR Mac(BSTR Message);
```

When the method is called, the responsible function in SCControl.ocx eventually zeroes a 1024-byte stack buffer and then copies the ”Message” argument passed to the method into it using an inline strcpy() call.

```
.text:10001C49                  mov     ecx, 100h
.text:10001C4E                  xor     eax, eax
.text:10001C50                  lea     edi, [esp+828h+szDest] ; char[1024]
.text:10001C54                  lea     edx, [esp+828h+szDest] ; char[1024]
.text:10001C58                  rep stosd
.text:10001C5A                  mov     ecx, 100h
.text:10001C5F                  lea     edi, [esp+828h+szBuf1] ; char[1024]
.text:10001C66                  rep stosd
.text:10001C68                  mov     edi, [esp+828h+bstrMessage]
.text:10001C6F                  or      ecx, 0FFFFFFFFh
.text:10001C72                  repne scasb
.text:10001C74                  not     ecx
.text:10001C76                  sub     edi, ecx
.text:10001C78                  push    offset aMac     ; "Mac"
.text:10001C7D                  mov     eax, ecx
.text:10001C7F                  mov     esi, edi
.text:10001C81                  mov     edi, edx
.text:10001C83                  mov     [esp+82Ch+var_4], 0
.text:10001C8E                  shr     ecx, 2
.text:10001C91                  rep movsd
.text:10001C93                  mov     ecx, eax
```

```
.text:10001C95                  and     ecx, 3
.text:10001C98                  rep movsb
```

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

Initialize() Method Handling Stack Buffer Overflow

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

```
[id(0x00000007)]
BSTR Initialize(BSTR Port);
```

When the method is called, the responsible function in SCControl.ocx eventually zeroes a 256-byte stack buffer and then copies the ”Port” argument passed to the method into it using an inline strcpy() call.

```
.text:10001DA9                  mov     ecx, 40h
.text:10001DAE                  xor     eax, eax
.text:10001DB0                  lea     edi, [esp+23Ch+Buffer]
.text:10001DB4                  lea     edx, [esp+23Ch+szDest] ; char[256]
.text:10001DBB                  rep stosd
.text:10001DBD                  mov     ecx, 40h
.text:10001DC2                  lea     edi, [esp+23Ch+szDest] ; char[256]
.text:10001DC9                  rep stosd
.text:10001DCB                  mov     edi, [esp+23Ch+bstrPort]
.text:10001DD2                  or      ecx, 0FFFFFFFFh
.text:10001DD5                  repne scasb
.text:10001DD7                  not     ecx
.text:10001DD9                  sub     edi, ecx
.text:10001DDB                  push    100h            ; uSize
.text:10001DE0                  mov     eax, ecx
.text:10001DE2                  mov     esi, edi
.text:10001DE4                  mov     edi, edx
.text:10001DE6                  mov     [esp+240h+var_4], 0
.text:10001DF1                  shr     ecx, 2
.text:10001DF4                  rep movsd
.text:10001DF6                  mov     ecx, eax
.text:10001DF8                  and     ecx, 3
.text:10001DFB                  rep movsb
```

As no bounds checks are performed, this may lead to 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-2019-010[1]
VulnDB:	202021, 202022, 202023, 202024, 202025, 202026, 202027

Timeline

2019-01-11	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-2019-010.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.