
RBS-2016-004

# Jensen of Scandinavia Air:Link Routers Web Interface Multiple Vulnerabilities

## Table of Contents

## Vendor / Product Information

Jensen of Scandinavia AS is a Norwegian company that develops networking and multimedia products and is a market leader in Scandinavia. As part of their networking product range is the popular Air:Link wireless router series, which is intended for both home and office use.

## Vulnerable Program Details

Details for tested products and versions:

| | |
|---|---|
| Vendor: | Jensen of Scandinavia AS |
| Product: | Air:Link 3G (AL3G) |
| Version: | 2.23m (Rev. 3) |
| | |
| Product: | Air:Link 5000AC (AL5000AC) |
| Version: | 1.13 |
| | |
| Product: | Air:Link 59300 (AL59300) |
| Version: | 1.04 (Rev. 4) |

NOTE: Other models and versions than the ones listed above are likely affected. The firmware is an OEM solution also used by other vendors including Belkin and Edimax, so devices from other vendors may also be vulnerable. In fact, it was found that some of the uncovered vulnerabilities were previously publicly disclosed in products from other vendors and fixed in some of these[1,2].

## Credits

Carsten Eiram, Risk Based Security
Twitter: @CarstenEiram
Twitter: @RiskBased

[1] http://blog.vectranetworks.com/blog/belkin-analysis
[2] http://seclists.org/bugtraq/2015/Dec/24

## Impact

Air:Link routers provide a web-based management interface for configuring the devices. The web service is based on GoAhead Webserver (webs) and runs with ‘root’ privileges. It was discovered that the web interface is affected by multiple shell command injection and stack-based buffer overflow vulnerabilities when accessing various /goform/ resources as an authenticated user. These allow execution of arbitrary code with ‘root’ privileges.

The devices are also affected by Cross-Site Request Forgery (CSRF) issues, which allow user-assisted exploitation of the previously mentioned vulnerabilities. Finally, a minor password disclosure and open redirect weaknesses were discovered.

## Vulnerability Details

### *__/goform/* Pages Stack-based Buffer Overflows__*

Many of the resources accessible in the /goform/ path are affected by stack-based buffer overflows when handling supplied parameter values. These resources are not actual CGI scripts, but implemented as procedures handled by the in-memory forms processor, GoForms, in the `/bin/webs` binary.

As these vulnerabilities are very similar in nature, only one of these, the /goform/mp resource, will be covered as an example.

When the resource is accessed, the `mp()` GoForms procedure within the `/bin/webs` binary is called, and the function retrieves the supplied ‘command’ parameter value.

```
.text:00457C3C mp:                                      # DATA XREF: main+E1Co
.text:00457C3C                                          # .got:mp_ptro
.text:00457C3C
.text:00457C3C var_218          = -0x218
.text:00457C3C szBuf1           = -0x210                # char[504]
.text:00457C3C var_18           = -0x18
.text:00457C3C var_14           = -0x14
.text:00457C3C var_10           = -0x10
.text:00457C3C var_C            = -0xC
.text:00457C3C var_8            = -8
.text:00457C3C
.text:00457C3C                  li      $gp, 0x5D004
.text:00457C44                  addu    $gp, $t9
.text:00457C48                  addiu   $sp, -0x228
.text:00457C4C                  sw      $ra, 0x228+var_8($sp)
.text:00457C50                  sw      $s3, 0x228+var_C($sp)
.text:00457C54                  sw      $s2, 0x228+var_10($sp)
```

```
.text:00457C58                  sw      $s1, 0x228+var_14($sp)
.text:00457C5C                  sw      $s0, 0x228+var_18($sp)
.text:00457C60                  sw      $gp, 0x228+var_218($sp)
.text:00457C64                  la      $s3, aTypeTextJavasc
.text:00457C68                  li      $a1, 0x470000
.text:00457C6C                  la      $t9, websGetVar
.text:00457C70                  addiu   $a2, $s3, (asc_45A164+4 - 0x460000)  # ""
.text:00457C74                  addiu   $a1, (aCommand - 0x470000)  # "command"
.text:00457C78                  jalr    $t9 ; websGetVar
.text:00457C7C                  move    $s0, $a0
.text:00457C80                  lw      $gp, 0x228+var_218($sp)
.text:00457C84                  move    $a0, $s0
.text:00457C88                  la      $t9, websHeader
.text:00457C8C                  nop
.text:00457C90                  jalr    $t9 ; websHeader
.text:00457C94                  move    $s2, $v0
```

The function eventually passes the value to the `sprintf()` function as argument when attempting to construct the "*/bin/rftest.sh %s > /tmp/rftest.out*" string into a fixed-size stack buffer. As no boundary checks are performed, this may lead to a stack-based buffer overflow.

```
.text:00457EA0                  li      $a1, 0x470000
.text:00457EA4                  jalr    $t9 ; websWrite
.text:00457EA8                  addiu   $a1, (aFontColorRedSF+0x14 - 0x470000)  #
"</font><br>\n"
.text:00457EAC                  lw      $gp, 0x228+var_218($sp)
.text:00457EB0                  move    $a2, $s2
.text:00457EB4                  li      $a1, 0x470000
.text:00457EB8                  la      $t9, sprintf
.text:00457EBC                  addiu   $a1, (aBinRftest_shST - 0x470000)  # "/bin/rftest.sh %s
> /tmp/rftest.out"
.text:00457EC0                  jalr    $t9 ; sprintf    # b0f!
.text:00457EC4                  addiu   $a0, $sp, 0x228+szBuf1  # char *
```

Similar vulnerabilities exist for many other resources. The following table lists all affected resources and parameters.

| Affected /goform/* Resource | Affected Parameter(s) |
|---|---|
| /goform/mp | command |
| /goform/formSysCmd | sysCmd |
| /goform/formUSBStorage | sub_dir |
| /goform/TestTTCP | localip, host, length, and number |
| /goform/formConnectionSetting | max_Conn and timeOut |
| /goform/formWpsStart | pinCode |

| | |
|---|---|
| /goform/formWlanMP | ateFunc, ateGain, ateTxCount, ateChan, ateRate, ateMacID, e2pTxPower1, e2pTxPower2, e2pTxPower3, e2pTxPower4, e2pTxPower5, e2pTxPower6, e2pTxPower7, e2pTx2Power1, e2pTx2Power2, e2pTx2Power3, e2pTx2Power4, e2pTx2Power5, e2pTx2Power6, e2pTx2Power7, ateTxFreqOffset, ateMode, ateBW, ateAntenna, e2pTxFreqOffset, e2pTxPwDeltaB, e2pTxPwDeltaG, e2pTxPwDeltaMix, e2pTxPwDeltaN, and readE2P |
| /goform/formHwSet | Anntena, Mcs, regDomain, nic0Addr, nic1Addr, wlanAddr, wanAddr, wlanSSID, wlanChan, comd, initgain, txcck, and txofdm |

### ***/goform/* Pages Shell Command Injection***

Should exploitation of the stack-based buffer overflows require too much effort, affected devices provide a simpler approach to execute code. All of the resources affected by the buffer overflows are similarly affected by shell command injection. These vulnerabilities are also very similar in nature. Only the `formAccept()` GoForms procedure will be discussed as an example.

When the /goform/formAccept resource is accessed, the associated `formAccept()` GoForms procedure in the `/bin/webs` binary is called. The function retrieves the supplied ‘submit-url’ parameter value and passes it straight to the `system()` C library function without any validation.

```
.text:00453D30 formAccept:                             # DATA XREF: main+D90o
.text:00453D30                                         # .got:formAccept_ptro
.text:00453D30
.text:00453D30 var_18          = -0x18
.text:00453D30 var_10          = -0x10
.text:00453D30 var_C           = -0xC
.text:00453D30 var_8           = -8
.text:00453D30
.text:00453D30                 li      $gp, 0x60F10
.text:00453D38                 addu    $gp, $t9
.text:00453D3C                 addiu   $sp, -0x28
.text:00453D40                 sw      $ra, 0x28+var_8($sp)
.text:00453D44                 sw      $s1, 0x28+var_C($sp)
.text:00453D48                 sw      $s0, 0x28+var_10($sp)
.text:00453D4C                 sw      $gp, 0x28+var_18($sp)
.text:00453D50                 la      $a1, aTypeTextJavasc
```

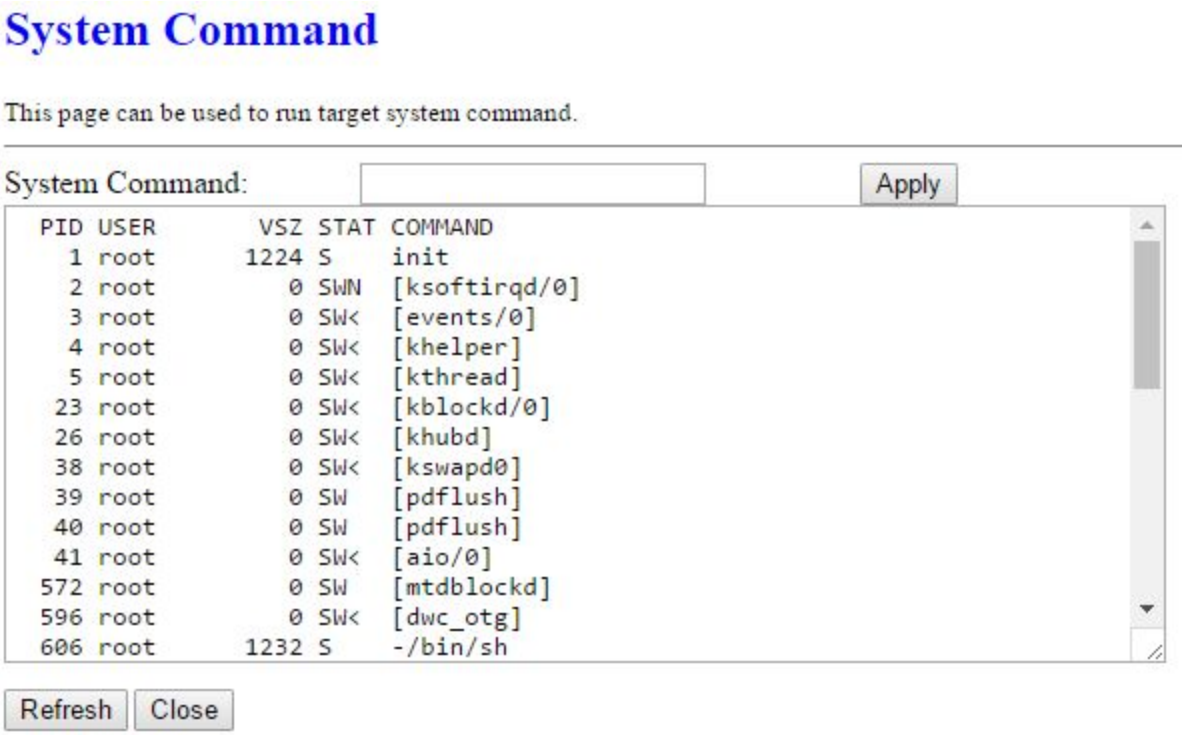
```
.text:00453D54                la      $a2, aTypeTextJavasc
.text:00453D58                la      $t9, websGetVar
.text:00453D5C                addiu   $a1, (aSubmitUrl - 0x460000)  # "submit-url"
.text:00453D60                addiu   $a2, (asc_45A164+4 - 0x460000)  # ""
.text:00453D64                jalr    $t9 ; websGetVar
.text:00453D68                move    $s1, $a0
.text:00453D6C                lw      $gp, 0x28+var_18($sp)
.text:00453D70                move    $a0, $v0          # string
.text:00453D74                la      $t9, system
.text:00453D78                nop
.text:00453D7C                jalr    $t9 ; system      # shell cmd injection!
```

This allows injecting and executing arbitrary shell commands with ‘root’ privileges. The following table lists all affected GoForms resources and parameters.

| Affected /goform/* Resource | Affected Parameter(s) |
|---|---|
| /goform/formAccept | submit-url |
| /goform/mp | command |
| /goform/formSysCmd | sysCmd |
| /goform/formUSBStorage | sub_dir |
| /goform/TestTTCP | localip, host, length, and number |
| /goform/formConnectionSetting | max_Conn and timeOut |
| /goform/formWpsStart | pinCode |
| /goform/formWlanMP | ateFunc, ateGain, ateTxCount, ateChan, ateRate, ateMacID, e2pTxPower1, e2pTxPower2, e2pTxPower3, e2pTxPower4, e2pTxPower5, e2pTxPower6, e2pTxPower7, e2pTx2Power1, e2pTx2Power2, e2pTx2Power3, e2pTx2Power4, e2pTx2Power5, e2pTx2Power6, e2pTx2Power7, ateTxFreqOffset, ateMode, ateBW, ateAntenna, e2pTxFreqOffset, e2pTxPwDeltaB, e2pTxPwDeltaG, e2pTxPwDeltaMix, e2pTxPwDeltaN, and readE2P |
| /goform/formHwSet | Anntena, Mcs, regDomain, nic0Addr, nic1Addr, wlanAddr, wanAddr, wlanSSID, wlanChan, comd, initgain, txcck, and txofdm |

Access to the /goform/formSysCmd resource is also possible in a simpler fashion by navigating

to the hidden `http://[IP]/syscmd.asp` web page. This provides a basic interface to run arbitrary commands on the device with 'root' privileges and view the output.




### *[/goform/* Pages Settings Manipulation CSRF]*

Multiple Jensen Air:Link devices contain a flaw in the web-based management interface, as HTTP requests to /goform/ pages do not require multiple steps, explicit confirmation, or a unique token when performing certain sensitive actions. By tricking a user into following a specially crafted link, a context-dependent attacker can perform a Cross-Site Request Forgery (CSRF / XSRF) attack causing the victim to manipulate device settings.

The following example restarts the device by combining one of the shell command injection vulnerabilities:
`http://[IP]/goform/formAccept?submit-url=reboot;`

### *[/x.asp User Credentials Disclosure Weakness]*

In order for an administrative user to change the password for the web-based management interface, the user has to re-authenticate by supplying the current password along with the new password. This is a common defense-in-depth security precaution to e.g. prevent attackers, who

somehow gained unauthorized access to a session, to change the password.

This defense-in-depth security precaution is ineffective for affected devices. The hidden `x.asp` page allows an authenticated attacker to view the current 'admin' account password and (hardcoded) 'super' account password.


It should be noted that it is not possible to log into the web-based management interface using the 'super' account credentials.

### *`/goform/` Pages submit-url Parameter Open Redirect Weaknesses*

45 different GoForms resources e.g. /goform/formStats, /goform/formAccept, and /goform/formrefresh accept the 'submit-url' parameter and redirect to the provided URL argument without any restrictions.

As these weaknesses are all similar in nature, only the `formStats()` GoForms procedure will be discussed as an example.

Upon accessing the /goform/formStats resource, the associated function within the `/bin/webs` binary is called. The value supplied to the 'submit-url' parameter is retrieved and passed as argument to the `websRedirect()` function provided by the GoAhead native API. This results in a 302 HTTP response being returned to the client with the 'Location' header set to the specified URL. No checks are performed to limit redirection to only resources on the same domain.

```
.text:00457808 formStats:                              # DATA XREF: main+E00o
.text:00457808                                         # .got:formStats_ptro
.text:00457808
.text:00457808 var_10          = -0x10
.text:00457808 var_8           = -8
.text:00457808 var_4           = -4
.text:00457808
.text:00457808                 li      $gp, 0x5D438
.text:00457810                 addu    $gp, $t9
.text:00457814                 addiu   $sp, -0x20
.text:00457818                 sw      $ra, 0x20+var_4($sp)
.text:0045781C                 sw      $s0, 0x20+var_8($sp)
```

```
.text:00457820                 sw      $gp, 0x20+var_10($sp)
.text:00457824                 la      $a1, aTypeTextJavasc  #
"type='text/javascript'>document.write(s"...
.text:00457828                 la      $a2, aTypeTextJavasc  #
"type='text/javascript'>document.write(s"...
.text:0045782C                 la      $t9, websGetVar
.text:00457830                 addiu   $a1, (aSubmitUrl - 0x460000)  # "submit-url"
.text:00457834                 addiu   $a2, (asc_45A164+4 - 0x460000)  # ""
.text:00457838                 jalr    $t9 ; websGetVar
.text:0045783C                 move    $s0, $a0
.text:00457840                 move    $a1, $v0
.text:00457844                 lw      $gp, 0x20+var_10($sp)
.text:00457848                 lb      $v0, 0($v0)
.text:0045784C                 la      $t9, websRedirect
.text:00457850                 beqz    $v0, loc_457868
.text:00457854                 move    $a0, $s0
.text:00457858                 lw      $ra, 0x20+var_4($sp)
.text:0045785C                 lw      $s0, 0x20+var_8($sp)
.text:00457860                 jr      $t9 ; websRedirect
```

This allows an attacker to create a specially crafted link that, if followed by a victim, would redirect from the intended legitimate web interface to an arbitrary web site of the attacker's choosing. Such attacks are useful as the crafted URL initially appears to be a web page of a trusted site. This could be leveraged to e.g. conduct phishing attacks that mimic the legitimate site, but send user-supplied information to the attacker.

Example:
`http://[IP]/goform/formStats?submit-url=http://[malicious_site]`

### *<u>/goform/formLogout return-url Parameter Open Redirect Weakness</u>*

Similar to how the ‘submit-url’ permits open redirects for many GoForms resources, the /goform/formLogout resource is affected when handling the ‘return-url’ parameter.

```
.text:00455F78 formLogout:                             # DATA XREF: main+71Co
.text:00455F78                                         # .got:formLogout_ptro
.text:00455F78
.text:00455F78 var_18          = -0x18
.text:00455F78 var_10          = -0x10
.text:00455F78 var_C           = -0xC
.text:00455F78 var_8           = -8
.text:00455F78
.text:00455F78                 li      $gp, 0x5ECC8
.text:00455F80                 addu    $gp, $t9
.text:00455F84                 addiu   $sp, -0x28
…
.text:00455FA0                 la      $t9, websGetVar
.text:00455FA4                 addiu   $a1, (aLogout - 0x460000)  # "logout"
.text:00455FA8                 addiu   $a2, $s0, (asc_45A164+4 - 0x460000)  # ""
.text:00455FAC                 jalr    $t9 ; websGetVar
```

```
.text:00455FB0                  move    $s1, $a0
.text:00455FB4                  lw      $gp, 0x28+var_18($sp)
.text:00455FB8                  lb      $v0, 0($v0)
.text:00455FBC                  la      $a1, aTypeTextJavasc
.text:00455FC0                  la      $t9, websGetVar
.text:00455FC4                  addiu   $a1, (aReturnUrl - 0x460000)  # "return-url"
.text:00455FC8                  addiu   $a2, $s0, (asc_45A164+4 - 0x460000)  # ""
.text:00455FCC                  beqz    $v0, loc_455FE0
…
.text:00455FE0 loc_455FE0:                              # CODE XREF: formLogout+54j
.text:00455FE0                  jalr    $t9 ; websGetVar
.text:00455FE4                  nop
.text:00455FE8                  lw      $gp, 0x28+var_18($sp)
.text:00455FEC                  move    $a0, $s1
.text:00455FF0                  la      $t9, websRedirect
.text:00455FF4                  nop
.text:00455FF8                  jalr    $t9 ; websRedirect
.text:00455FFC                  move    $a1, $v0
```

Similar to the weaknesses discussed above, this may be exploited to e.g. conduct phishing attacks.

Example:
`http://[IP]/goform/formLogout?return-url=http://[malicious_site]`

### *Other Potential Issues*

It should be noted that other vulnerabilities including XSS (Cross-Site Scripting) and HTTP header injection[3] were previously reported by other researchers in products from other vendors using the same or similar OEM firmware.

These issues likely also affect the discussed Air:Link devices from Jensen of Scandinavia. However, after having uncovered 40+ distinct vulnerabilities, we decided to terminate our analysis. More issues are likely to be found. Clearly the vendor should invest in a thorough security assessment of their products.

[3] http://www.s3cur1ty.de/node/673

## Solution

The vendor was unresponsive when contacted. We are not currently aware of a solution for these vulnerabilities. As the vendor advertises a 20 year guarantee on their devices[4], we encourage customers to make use of this by either requesting a refund or demanding that the vulnerabilities are addressed in a timely manner.

## References

| | |
|---|---|
| RBS: | RBS-2016-004[5] |
| VulnDB IDs: | 148933, 148932, 148936, 148945, 148937, 148938, 148941, 148934, 148935, 148940, 148939, 148942, 148943, 148944, 148951, 148950, 148949, 148948, 148947, 148946 |

## Timeline

| | |
|---|---|
| 2016-11-28 | Vulnerabilities discovered. |
| 2016-11-29 | Vulnerabilities reported to the vendor |
| 2016-12-19 | No vendor response. Alerts sent to RBS VulnDB clients. |
| 2016-12-29 | Publication of this vulnerability report. |

[4] http://www.jensenofscandinavia.com/en/warranty/
[5] https://www.riskbasedsecurity.com/research/RBS-2016-004.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, 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.