Hack the legacy!

*IBM i (aka AS/400)* revealed.

Bart Kulach
Agenda

- Let’s get introduced
- Why should we care about legacy?
- Evil Java?
- Privilege escalation – let’s jump!
- Password security and hash grabbing
- Summary + Q&A
Let’s get introduced

🌟 I’m googleable.

https://nl.linkedin.com/in/bartkulach

Disclaimer:

Any views or opinions presented here are solely those of the author and do not necessarily represent those of his employer(s).
Why should we care about legacy?
Why should we care about legacy?

- It’s legacy… but hard to get rid of.
- It’s processing most interesting data.
- It’s usually less secure than front-ends.
- It’s often more vulnerable than you think.
- It’s still quite accessible to potential intruders.
- It’s existing everywhere - in all economic sectors.
- It’s already been exploited ("Hacking iSeries" by S.Carmel)!
Evil Java?
Evil Java?

- IBM Toolbox for Java/JTOpen
- Allows for remote system API calls and usage of built-in system commands ("Limited capability" not effective here)
- Gives the flexibility of coding "outside" the AS/400 box (no need for extra authorities on the system)
- Is generally poorly written (decompile and check yourself!)
- Handling of authorisations by Java VM on server side is inconsistent (object authority vs. data authority), allowing for greater visibility
Demo time:
Evil Java – visibility example
Privilege escalation – let’s jump!
Privilege escalation – let’s jump
Part 1 – remote profile switching

- Do you use group profiles? Like one common group profile?
- Are your admins also members of the group?
- Are your object and data authorities hardened?
- Do you monitor profile handle swapping?

Let’s jump remotely:
- check the list of profiles you have access to
- grab a profile handle
- switch to the profile
- repeat until you’re happy with your access level 😊
Demo time:
Remote profile switching
Exit points/programs generally allow to protect the system quite easily from usage of specific SQL queries or system commands.

Most commercial protection software that use exit programs have their weaknesses/vulnerabilities.

They can be however often be circumvented by using nested commands (commands running commands).

Especially if you cross the environments (CL–PASE–DB2)...

And if we add JDBC to that... like

\[
\text{CALL QSYS.QCMDEXEC('QSH CMD(''DB2 \"select * from library.file\" | Rfile -w /QSYS.LIB/QSYSPRT.FILE'')', 0000000077.000000)}
\]
Demo time: Nested command use
Password security and hash grabbing
Password security and hash grabbing

- IBM offers you a nice API (QSYRUPWD) to grab the hashes.

- QSYRUPWD allows for getting an extract of all hashes for a particular user.

- The output format is proprietary and was never published until today 😊

- Is your QPWDLVL system value 0, 1 or 2*? If so, you can enjoy the LM hashes 😊
  *
  for QPWDLVL = 2, QPWDMAXLEN must be <=14

- You have to be *SECADM (and ideally *ALLOBJ) though (so go back and escalate your privileges first).
Password security and hash grabbing – cont’d.

Retrieve Encrypted User Password (QSYRUPWD) API

Required Parameter Group:
1. Receiver variable
2. Length of receiver variable
3. Format
4. User profile name
5. Error code

Default Public Authority: *EXCLUDE

Threadsafe: No

UPWD0100 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Field</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>CHAR(*)</td>
</tr>
</tbody>
</table>
## Password security and hash grabbing – cont’d.

**QSYRUPWD** Encrypted password data hex string

<table>
<thead>
<tr>
<th>Offset (Dec)</th>
<th>Length (Chars)</th>
<th>Field</th>
<th>QPWDLVL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>DES 56-bit encrypted password substitute (RFC2877)</td>
<td>0, 1, 2*</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>DES 56-bit encrypted password substitute (RFC2877)</td>
<td>0, 1, 2*</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>LM hash</td>
<td>0, 1, 2*</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>No data</td>
<td>-</td>
</tr>
<tr>
<td>68</td>
<td>40</td>
<td>HMAC-SHA1 encrypted password token (RFC4777)?</td>
<td>0**, 1**, 2, 3</td>
</tr>
<tr>
<td>108</td>
<td>40</td>
<td>HMAC-SHA1 encrypted password token (RFC4777)?</td>
<td>0**, 1**, 2, 3</td>
</tr>
<tr>
<td>148</td>
<td>6</td>
<td>No data</td>
<td>-</td>
</tr>
<tr>
<td>154</td>
<td>384</td>
<td>Unknown (hash?) data</td>
<td>0, 1, 2, 3</td>
</tr>
</tbody>
</table>

*depending on password rules; **from V5R1 onwards*
Demo time:
Password grabbing
Summary + Q&A

- Java is the evil for AS/400.
- Be sceptic about IBM Security books.
- Visit www.hackthelegacy.org
@bartholozz
www.hackthelegacy.org